1 Overview and Parameters

This program is designed to compute amplitude density, autospectral (power) density and cross-spectral density functions for discrete, stationary random time series in the form

$$x_n = x\,(t_0 + n\Delta t) \qquad \text{for } n = 0, 1, 2, \ldots, N-1. \tag{1.1}$$

where N is the number of data points per record. Typically, M such records are stored in a data file, and each record is a contiguous segment of the stationary process. Most often, this data is in the form of 2-byte integers after having been digitized by an analog to digital converter. This routine reads this data file and then transforms the integer data back into voltages or optionally into some other desired quantity. This transformation into a dimensional quantity is by means of a polynomial of up to fourth degree. The coefficients necessary to complete the transformation are placed in coefficient data files which are read by the routine. Alternatively, the routine can read 4-byte floating-point data which has already been transformed by some other program. The input data may consist of one or two separate time series which are to be analyzed.

After transformation, various cosmetic operations may be performed on the data prior to the spectrum computation. The mean value, which is calculated during the transformation process, may be subtracted from the data; or instead, a linear trend in the data may be removed using a least squares fit technique. Recursive digital filters including lowpass, highpass, bandpass and bandstop have been incorporated and may be applied prior to the calculation of the spectra. Finally, to reduce the leakage of power from other frequencies due to the fact that the input time series are of finite length and are then truncated, a tapering operation has been included. A choice of four tapering functions is supplied, and the appropriate scale factors are computed automatically. The use of tapering may introduce some additional variability in the resulting spectral estimate; to counteract this tendency, a means for overlapping the input data records is provided. This option employs 50% overlap to reduce overall error at the expense of computation time (which is doubled); however, the use of overlapping requires that the data be contiguous in time from record to record.

After computation of the selected spectral density function, the data are stored to an ASCII output file which may then be imported into a plotting program. The routine is designed for multiple file processing; many input data files may be processed sequentially if certain naming conventions are followed. The maximum number of files is currently set to IFMAX = 100 in order to allocate storage for data. Entering data file names on the command line upon invocation of the program is also permitted and the maximum is set to JFMAX = 5. Each record of the input data file may contain no more than NMAX = 16384 points; note that in the case of two channels of data per record, each channel may contain no more than NMAX / 2 points. As described below, the order of the recursive digital filters is set by the number of cascaded stages desired; the maximum number of stages is currently placed at NSMAX = 20. For convenience, the base 10 logarithms of certain data in the

output are computed; to avoid taking the logarithm of zero, any number in the output data less than $\varepsilon = 1.0\,x\,10^{-20}$ is set equal to ε. Any of these maximum values may be altered by making a change in the appropriate PARAMETER statement near the beginning of the code and then recompiling.

The speed of the routine is limited by the large number of I/O operations that are required; reduce the number of points per record and/or the number of records per file for increased speed. Alternatively, specify in the configuration file that the routine use a RAM disk for storage of temporary files. The size of the executable file is largely governed by the setting of NMAX; since this parameter must be a power of two, the largest possible value is 16384 for a DOS-based computer which results in an executable file size of 430K bytes. The CHKDSK command must report that at least 450K bytes of memory are present for the routine to execute for this value of NMAX. When compiling the source code using Microsoft Fortran V 5.00, at least 603K of RAM must be available for full optimization of the code.

When the data to be processed by the spectrum routine are samples of a continuous function of time at a single spatial location, then the calculated spectral density functions are a function of frequency. The user should keep in mind that if the input data are samples of a continuous function of one spatial direction at an instant of time (such as intensity data for horizontal or vertical rows of pixels on a photograph from a CCD camera), then this routine can be used to determine one-dimensional estimates of wavenumber spectra.

2 Input Data Files

For any routine in which the processing of large amounts of data is required, a rather rigid structure is necessarily imposed on the manner in which the data is transferred into the program for manipulation. The spectrum routine is certainly no exception to this rule. Furthermore, since the spectrum routine has the ability to process data files in batches, additional rules are required, such as naming conventions, to allow this feature to be implemented. Although the proliferation of rules can be a bit wearying, later sections discuss some situations in which the rules can be bent or discarded.

2.1 Naming Convention

A convention for naming the input data files was chosen to facilitate multiple file processing. The selected method resulted from the ease with which the file names could be generated by a few lines of code. Each file name consists of eight characters: the first character must be alphabetic (A-Z), the next three characters are digits (0-9) and the remaining four characters are a period followed by the letters DAT. Note that the routine is not case-sensitive, lower-case as well as upper-case characters may be entered; however, all lower-case inputs are converted to upper-case internally. The routine automatically appends the suffix [.DAT] and therefore it should never be entered upon input. The data file must reside in the same directory as the executable file unless specified otherwise in the configuration file (discussed later). Examples of acceptable data file names are: A001.DAT, z087.dat and M418.DAT. The latter filename would require that IFMAX $\geq$ 418. Every rule has an exception; data file names entered on the command line when calling the routine need not be so restrictive as outlined above. They must still consist of eight characters and the first character must still be alphabetic (A-Z); however, the next three characters can be any of the alphanumeric characters. The blank character is used to delimit the names on the command line and is therefore not a valid character for a file name. The remaining four characters must be a period followed by the letters DAT. Examples of acceptable filenames for files entered on the command line are: TEST.DAT, x_in.dat and S1$$.DAT. When actually entering the names on the command line, only the first four characters are entered and the program appends [.DAT].

2.2 File Structure

All input data files must be FORTRAN data files opened as sequential, unformatted files. The minimal amount of internal formatting allows the files to be more compact which is a consideration when dealing with large quantities of data.

2.2.1 File Header

The first record of the data file consists of a header containing four quantities which characterize the data which follows. The first quantity is a 2-byte integer which gives the number of channels of data per record (1 or 2). The next number is the record size in bytes and may be either a 2-byte or a 4-byte integer. The record size is twice the number of points per record if the data consists of 2-byte integer values, or the record size is four times the

number of points per record if the data consists of 4-byte real data. The third quantity is a 2-byte integer which gives the number of records of data which follow, and the last quantity is the time interval *in microseconds* between data points *of the same channel* expressed as a 2-byte integer. Recall that the largest positive signed number that may be stored as a 2-byte integer is 32,767 and, for a 4-byte integer, the value is 2,147,483,647.

For example, suppose the data file consists of 100 records of one channel of data saved as 2-byte integers and sampled at 10000 samples per second with 2048 data points per record. The file header would contain the four values:

1 4096 100 100

Now suppose that two channels of data were sampled with a time interval of 0.016383 seconds between each successive data point resulting in a time interval of 0.032766 seconds between each successive data point *of the same channel*. Furthermore, 40 records of data were sampled with 8192 data points per record such that each channel of data consisted of 4096 data points. Finally, the data were transformed into desired quantities and saved as 4-byte floating-point numbers. The header for this data file should consist of the following four numbers:

2 32768 40 32766

2.2.2 Remainder of Data File

Following the first record containing the file header is the actual data. When the data comes directly from an analog to digital converter (12-bit, say), then the values range from -2048 to 2047 (or 0 to 4095) and are appropriately stored as 2-byte integers. Alternatively, if some preprocessing is typically performed on the data, the data may be stored as 4-byte floating-point numbers. As a result of the Fast Fourier Transform (FFT) algorithm used in this routine, the number of data points per channel per record *must be a power of 2*. One or two separate time series may be stored per record. If only one channel of data is sampled then *an even number of records* following the file header should be stored in the data file. This is necessary because in the case of one channel of data, two records are submitted to the FFT algorithm to be transformed simultaneously as a time-saving measure. If an odd number of records is detected, then the program automatically appends an additional record of data containing a copy of the data values of the previous record. When two channels of data are sampled, the number of records may be even or odd. The method of storing the data values in each record follows the standard procedure used by analog to digital converters:

one channel: $x_0\, x_1\, x_2\ \ldots\ x_{N-1}$

two channels: $x_0\, y_0\, x_1\, y_1\, x_2\, y_2\ \ldots\ x_{N-1}\, y_{N-1}$

where the notation is as defined above. This convention must be followed even if the data are processed and then stored as floating-point numbers. When two channels of data are sampled and then stored to a data file, the values for each channel of data should be of the same data type: either 2-byte integers or 4-byte floating-point numbers. If the user is unfamiliar with the task of creating unformatted FORTRAN data files, then the user may instead create these files in an ASCII file format. A utility MAKDAT which is included on the diskette can then be used to write the file header and transform the ASCII data files into the unformatted FORTRAN data files required by the spectrum routine. See the section on utility files.

2.2.3 Writing and reading the Data File

This section gives a few lines of code which are representative of the manner in which these data files are written and read. When writing, the file is first opened, the file header is written, the data is written and then the file is closed.

```
INTEGER*2 ICHANS, IRSIZE, NUMREC, IDELTMS
INTEGER*2 NDATA (NMAX)

OPEN (2, FILE = 'A001.DAT', STATUS = 'UNKNOWN',
     ACCESS = 'SEQUENTIAL', FORM = 'UNFORMATTED')

WRITE (2) ICHANS, IRSIZE, NUMREC, IDELTMS

DO J = 1, NUMREC
  ... put data record into NDATA array ...
  WRITE (2) (NDATA (I), I = 1, N)
  ... get next record of data ...
ENDDO

CLOSE (2, STATUS = 'KEEP')
```

The same procedure is followed when the data file is opened and read by the spectrum routine. The file is opened, the file header is read, the data is read and then the routine is closed.

```
INTEGER*2 ICHANS, IRSIZE, NUMREC, IDELTMS
INTEGER*2 NDATA (NMAX)

OPEN (2, FILE = 'A001.DAT', STATUS = 'OLD',
     ACCESS = 'SEQUENTIAL', FORM = 'UNFORMATTED')

READ (2) ICHANS, IRSIZE, NUMREC, IDELTMS

N = IRSIZE/2
DELT = FLOAT (IDELTMS) / 1000000.0

DO J = 1, NUMREC
  READ (2) (NDATA (I), I = 1, N)
   ... process this record of data ...
ENDDO

CLOSE (2, STATUS = 'KEEP')
```

2.3 Multiple File Processing

This routine is capable of sequentially processing many files in a batch. The program will handle up to IFMAX files which are consecutively numbered or which are not consecutively numbered but have an arbitrary numbering arrangement. All data files in the batch must start with the same first letter. The letter and number combination in the file name are used to associate each data file to a corresponding set of coefficients (if needed) in an accompanying coefficient data file; this is the reason for the rather rigid naming convention. Alternatively, up to JFMAX files may be entered on the command line when calling the program, and these will be processed sequentially. These files need not have the same first letter; however, these data files also cannot use coefficient data files. This is further clarified below. All data files must conform to the naming convention described above. The selection of options to perform various operations on the data will then be applied to all files in the batch.

2.3.1 Consecutively numbered files

If it is desired to process several files in a consecutive order, then this option is selected. The routine will ask for the beginning and ending file numbers and then the first letter in the file names. As an example, suppose it was desired to process files C031.DAT through C094.DAT, then the user would respond as follows:

```
spectrum

Process files (C)onsecutively or in an (A)rbitrary order : c
```

Enter starting & ending data file # : 31 94 (or 31,94)

Enter first letter of data file : c

2.3.2 Arbitrarily numbered files

Perhaps it is not desirable to process all of the files indicated above, but only selected ones from the set. If it were necessary to process C031.DAT, C042.DAT, C067.DAT and C085.DAT, then the user would respond as follows.

SPECTRUM

Process files (C)onsecutively or in an (A)rbitrary order : A

Enter number of files to process : 4

Enter last digits of file 1 : 31
Enter last digits of file 2 : 42
Enter last digits of file 3 : 67
Enter last digits of file 4 : 85

Files are :
 31 42 67 85 (routine echoes file numbers selected)

Enter first letter of data file : C

2.3.3 Files entered on command line

This option was added as a means of processing a few files without having to answer all of the prompts above and to allow a bit more freedom in the selection of file names. Suppose it was desired to process TSTX.DAT, X_IN.DAT and WOW!.DAT, then the user would enter the following command line.

SPECTRUM TSTX X_IN WOW!

By relaxing the naming convention, it is no longer possible to easily associate a coefficient data file with the correct input data file. Therefore, if the user desires to transform the integer data in each of the input files to voltages only, or if the input data in each file already consists of transformed floating-point numbers (as a result of some other program), then coefficient data files are not necessary and this method for quickly processing a small number of files may be used. If the integer data in each input file must be transformed into a desired floating-point quantity using a polynomial transformation prior to the spectral calculation,

then the program must be initiated using the methods described in sections 2.3.1 or 2.3.2. Further discussion on the manner in which coefficient data files are associated with the respective input data files appears below.

2.4 Coefficient Data Files

These data files may be sequential unformatted or sequential formatted FORTRAN data files. They contain the constants which are used as the coefficients of the polynomial that transforms the input data from voltages to some other desired quantity.

2.4.1 Naming Convention

The coefficient data file names consist of eight characters. The first letter must be alphabetic (A-Z) and must be the same letter as the input data file (or series of input data files) with which it is associated. The spectrum routine assigns the remaining seven characters of the name as CON.DAT (if the file is unformatted) or CON.PRN (if the file is formatted). If the input data file D061.DAT were to be transformed from voltages into velocities, then the coefficients of the polynomial would be found in the coefficient data file named DCON.DAT (if unformatted) or DCON.PRN (if formatted), and this file must reside in the same directory as that of the executable routine and the input data file unless specified otherwise in the configuration file (discussed later). If D061.DAT contained two channels of data, then the coefficients needed to transform the data for the second channel would be found in the coefficient data file named DCON2.DAT (if unformatted) or DCON2.PRN (if formatted). Again, the routine requests that the user enter the first letter only and the remaining eight characters are automatically assigned by the routine.

2.4.2 Unformatted File Structure

These coefficient data files are accessed only if the input data are 2-byte integers and the user desires to convert the integers first into voltages and then into another quantity. The voltage transformation factor is not included in these files; instead, it is entered into the routine at the beginning and therefore applies to all files processed. If a series of files are to be processed, the coefficients for the transformation from voltages into the other quantity may not be the same for all files in the batch, and this situation is addressed as follows. The conversion from a voltage into a velocity, say, for each data point is carried out using

$$u_n = b_0 + b_1 v_n + b_2 v_n^2 + b_3 v_n^3 + b_4 v_n^4 \tag{2.1}$$

where v_n is the nth data point expressed as a voltage, $b_0, b_1, \ldots, b_4$ are the coefficients of the velocity transformation and u_n is the nth data point which has been converted to a velocity. The five coefficients of the transformation remain the same for each data point in each record of the file, but are permitted to vary from file to file. These constants are stored in a two-dimensional array of 4-byte floating-point numbers. The first index of the array is the number of the file to which the coefficients are to apply (1 to IFMAX), and the second

index is the number of the coefficient (1= b_0 to 5= b_4). The first record of the unformatted coefficient data file is a file header containing two 2-byte integer quantities: the first is the number of sets of coefficients to read, and the second is the starting file number to which these coefficients should be associated. The remainder of the data file consists of one large record containing all of the coefficients. If the user wishes to first create these coefficient data files in an ASCII file format (without the header), the utility MAKCOEF has been included on the diskette which will convert these ASCII files into the unformatted FORTRAN files needed by the spectrum routine. See the section on utility routines. An example of the code used to write and read a sequential unformatted coefficient data file follows.

```
INTEGER*2 NUMSETS, NUMCON, NSTART
REAL*4 CONST (IFMAX, 5)

NUMCON = 5

OPEN (2, FILE = 'DCON.DAT', STATUS = 'UNKNOWN',
    ACCESS = 'SEQUENTIAL', FORM = 'UNFORMATTED')

WRITE (2) NUMSETS, NSTART
WRITE (2) ((CONST (I,J), J = 1, NUMCON), I = 1, NUMSETS)

CLOSE (2, STATUS = 'KEEP')
```

The spectrum routine then reads the coefficient data file using code of the following form.

```
INTEGER*2 NUMSETS, NUMCON, NSTART
REAL*4 CONST (IFMAX, 5)

OPEN (2, FILE = 'DCON.DAT', STATUS = 'OLD',
    ACCESS = 'SEQUENTIAL', FORM = 'UNFORMATTED')

READ (2) NUMSETS, NSTART
READ (2) ((CONST (NSTART + I - 1, J), J = 1, NUMCON), I = 1, NUMSETS)

CLOSE (2, STATUS = 'KEEP')
```

The coefficients $b_0, b_1, \ldots, b_4$ for the data file D061.DAT are then given by CONST (61,1), CONST (61,2), ... , CONST (61,5).

2.4.3 Formatted Data Files

It is not always convenient to store coefficients in sequential unformatted data files. This option was included to allow a quick way of storing coefficients. A sequential formatted file is exactly that created by any simple text editor. To create a sequential formatted coefficient data file for the input data file D061.DAT, enter a text editor or word processor (capable of creating ASCII files) and give the file the name DCON.PRN. Next, enter the five coefficients, $b_0, b_1, \ldots, b_4$, one per line and then save the file. If you display the file to the screen, it should look something like the following. Note that the numbers should all be floating-point numbers.

3.14159265359	(b_0)
0.0	(b_1)
1.012345E-03	(b_2)
0.1	(b_3)
2.3E-05	(b_4)

An example of the code that the spectrum routine will use to read in this data file follows.

```
REAL*4 B (5), CONST (IFMAX, 5)

OPEN (2, FILE = 'DCON.PRN', STATUS = 'OLD')
 READ (2, *) (B (I), I = 1, 5)
CLOSE (2, STATUS = 'KEEP')

DO I = 1, IFMAX
 DO J = 1, 5
  CONST (I, J) = B (J)
 ENDDO
ENDDO
```

As is seen from the above code, the routine assigns the five constants to apply to all of the IFMAX different input data files that might be read. If a different set of constants is required for each data file, then the files must be processed one at a time, or sequential unformatted data files as described above must be used. If an input data file (D061.DAT, say) contains two channels of information, then a second sequential formatted data file with the name DCON2.PRN should be prepared as described above.

3 Data Preparation

This section deals with a sequence of *cosmetic* operations which may be applied to the data prior to performing spectral calculations. The first section discusses the transformation of the input data from integer values into voltages and then optionally into some other desired dimensional quantity (for example, velocities). If the input data consists of floating-point values, then the transformation operation is skipped. If desired, the mean value may be removed from each record of the input data, or a linear trend resulting, perhaps, from instrumentation drift may be removed from each record. Unwanted frequencies in the input data can be removed using the recursive digital filtering routines discussed below. Finally, a tapering operation which may be coupled with a 50% overlap of the input data records is included to eliminate the discontinuities found at the beginning and end of these records resulting from the fact that only a finite amount of data can be sampled. Currently, the user has a choice of four different window functions. If the input consists of two channels of data, then the first three operations: transformation, mean or trend removal and filtering may be different for each of the two input channels. The choice of whether or not to employ tapering as well as the choice of window function, however, applies to both input channels.

3.1 Transformation

Experimental data sampled by means of an analog-to-digital (A/D) converter typically results in data files containing an array of two-byte integer values. This is due to the fact that a two-byte integer is capable of representing numbers in a range from -32768 to 32767. This range is sufficient for 12, 14 and 16-bit converters and is used to approximate the magnitude of an analog voltage at a given instant of time. This voltage is constrained to lie within a specified input voltage range which may or may not be programmable and which is particular to the A/D converter used. Before proceeding to the spectral calculation, the integer input data must be transformed back into voltages using the scale factor appropriate for the input voltage range of the A/D converter used.

During development of this routine, a 12-bit A/D converter was used which was capable of digitizing three different bipolar voltage ranges: -1V to 1V, -5V to 5V and -10V to 10V. The use of 12 bits allows integer values ranging from -2048 to 2047 for a total of 4096 digital levels to be used to quantize each of the above input voltage ranges. The appropriate scale factors for conversion of the integer data back into voltages are given by

$$\frac{2\ V}{4096\ units}\ ,\quad \frac{10\ V}{4096\ units}\quad \text{and}\quad \frac{20\ V}{4096\ units}\ , \tag{3.1}$$

respectively. The user may choose from one of these scale factors or may enter an alternative scale factor from the keyboard. The routine first prompts the user with:

Transform into (V)oltages or into (O)ther quantity
or read (R)eal data which is already transformed :

If the input data consists of floating-point values, then the letter R should be entered, and the routine skips the rest of the transformation operation. For integer input data, the user responds with V for transformation into voltages, or O for voltage transformation followed by an additional transformation into a second dimensional quantity; the following prompt is then displayed:

Enter VOFST 1) 10/4096, 2) 20/4096, 3) 2/4096 or 4) other :

To this prompt, the user responds with a digit (1-4) indicating the desired scale factor for voltage transformation. Selection of item (4) leads to the prompt:

Enter VOFST :

and, at this point, the user enters a floating-point number representing the desired scale factor.

The input data will now be converted back into voltages using the scale factor specified above. Since the analog voltage sampled by the A/D converter often comes from a transducer, which measures some dimensional quantity and then converts it to a voltage signal, provision for transforming the digitized input voltages back into the original transduced quantity is included. Knowledge of calibration data for the particular transducer is required for this transformation, and the calibration curve may be linear, quadratic, exponential, etc. It was impossible during development of the code to anticipate all forms that the calibration curve might take; therefore, a polynomial fit was decided upon in hopes that it would satisfy the calibration requirements of most users. The polynomial may consist of terms up to fourth degree; however, lower degree polynomial representations may be used by setting the coefficient for the appropriate term to zero. The conversion from a voltage into a velocity, say, for each data point is carried out using

$$u_n = b_0 + b_1 v_n + b_2 v_n^2 + b_3 v_n^3 + b_4 v_n^4 \tag{3.2}$$

where b_0 through b_4 are the five calibration coefficients, v_n is the nth data point of the current record which has just been transformed into a velocity as described above and u_n is the resulting nth velocity value. The coefficients necessary to complete the transformation must be included in a suitable format in coefficient data files which are then read by the routine; instructions for the creation of these coefficient data files are contained in section 2.4.

An alternative to the transformation operations described herein is also available. The user may transform the original integer data from the A/D converter by means of his own program into floating-point data which can then be read by this routine. The floating-point input data files must conform to the structure set forth in section 2. The routine then skips the transformations described above and proceeds to the calculation of the mean and root-mean-square (rms) values of each data record and averaged values for the file.

Summarizing then, the spectrum routine begins by reading in an input data file. If this file consists of integers, then these integers are transformed to voltages using a voltage transformation factor (VOFST) selected by the user. Optionally, these voltages can be further transformed into some other dimensional quantity. If the input data file contains floating-point numbers, then the routine assumes that any transformation of the input data has already been applied by the user. As the input data is read in by the spectrum routine, record by record, the mean and rms values of each record are calculated and displayed on the screen at run-time. After all records of the input file have been read, the mean and rms values for each record are averaged to obtain values which are representative of the entire file. These averaged mean and rms quantities are also displayed on the screen at run-time. The spectrum routine then proceeds to the various data preparation operations discussed in the next section. Note that the averaged mean and rms values for the file are calculated a second time later on in the program as a result of the spectral computation and are also displayed on the screen. Theoretically, these two sets of values should be identical and serve as a check on the accuracy of the spectral computation. However, if any of the data preparation operations to be discussed next are applied to the data to improve the accuracy of the spectral computation, the two sets of mean and rms values may not exactly match. This is due to the fact that the data preparation operations *change the input data*. The first set of mean and rms values are calculated *prior* to the data preparation operations, and the second set are computed from spectral density calculations which occur *subsequent* to the data preparation procedures.

3.2 Mean Removal

For calculation of spectral quantities, it is common to remove the mean value from the input data. Both this operation and the trend removal operation described in the next section may be used for the removal of the mean from the sample data. Trend removal is not always desirable; for that reason a separate operation to remove the mean is needed and is described here. The mean value for each record of data in the input file is calculated from

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{3.3}$$

where x_n is defined as in equation 1.1; for convenience, and to reduce the algebra in the derivations in this section, n is defined such that $n = 1, 2, \ldots, N$, and the initial point of the record t_0 is set to zero. If y_n is used to represent the data with zero mean, then this quantity is computed from

$$y_n = x_n - \bar{x} \tag{3.4}$$

3.3 Removal of Linear Trend

As recommended by Bendat and Piersol (1986), a linear trend removal operation has been included in this routine. This operation is designed to remove low frequency components in the data with a period longer than the record length $T = N\Delta t$. These low frequency trends can significantly distort spectral data; however, trend removal should only be performed if instrumentation drift or other sources of trends are expected. Use of trend removal when no trends are present can introduce small contributions to the spectral density at frequencies on the order of the reciprocal of the record length.

This operation is accomplished by fitting a straight line through the data using the least squares fit method. The values calculated from this fit are then subtracted from the original data to yield data with zero mean and with no frequency components smaller than $1/T$ where T is the record length. Let $\tilde{x}$ be a first degree polynomial fit to the original data x_n (defined as in equation 1.1) where $\tilde{x}$ is given by

$$\tilde{x}_n = b_0 + b_1(n\Delta t) \qquad \text{for } n = 1, 2, \ldots, N \tag{3.5}$$

and with b_0 and b_1 defined as

$$b_0 = \frac{2(2N+1)\sum_{n=1}^{N} x_n - 6\sum_{n=1}^{N} n x_n}{N(N-1)} \tag{3.6}$$

$$b_1 = \frac{12\sum_{n=1}^{N} n x_n - 6(N+1)\sum_{n=1}^{N} x_n}{\Delta t\, N(N-1)(N+1)} \tag{3.7}$$

Let y_n represent the detrended data; these data are obtained from the following difference

$$y_n = x_n - \tilde{x}_n \tag{3.8}$$

3.4 Filtering

The filtering operations incorporated in this routine were added to allow the user to pass only desired frequencies in the input data on to the spectral calculation. Four recursive (infinite impulse response) Butterworth digital filters for filtering in the time domain have been included for this purpose: lowpass, highpass, bandpass and bandstop (notch). A short discussion of digital filtering in the time domain including a summary of the particular filters used in this routine and instructions for their use follows. Note however that filtering may

be applied not only in the time domain but also in the frequency domain. When filtering in the frequency domain, the input data record is Fast Fourier Transformed, then the FFT output is multiplied by a filter function, $H(f)$; if required, an inverse FFT could be performed to get the filtered data back into the time domain. Frequency domain filtering may perhaps be implemented in a future version of this program.

Consider a set of discrete input values x_n defined as in equation 1.1. Digital filtering involves a computing algorithm in which a set of filtered values y_n is produced as a result of operating on the inputs x_n . For the filters included in this routine, this is a linear operation. Digital filters can generally be categorized as recursive or nonrecursive; if the formula for y_n contains only input terms x_n , then the filter is termed nonrecursive. The general form of the linear nonrecursive algorithm is given by

$$y_n = \sum_{m=-M}^{M} b_m x_{n-m} \tag{3.9}$$

where b_m are the coefficients of the transformation. For values of $m < 0$, the above algorithm requires *future* values of the signal $x(t)$; this can cause a *realizibility* problem in analog systems, but can be accomplished in digital systems where, in many cases, the future values of $x(t)$ can be stored in advance. A recursive filter is one in which *previously filtered* values such as y_{n-1} , y_{n-2} , etc. appear explicitly in the algorithm for y_n . For recursive filters, the above equation is modified to include past values of the filtered output $y(t)$ as follows:

$$y_n = \sum_{m=-M}^{M} b_m x_{n-m} - \sum_{m=1}^{M} a_m y_{n-m} \tag{3.10}$$

The transfer function $H(f)$ for the filter is related to the coefficients a_m and b_m by

$$H(f) = \frac{\sum_{m=-M}^{M} b_m e^{-i2\pi f m\Delta t}}{1 - \sum_{m=1}^{M} a_m e^{-i2\pi f m\Delta t}} \tag{3.11}$$

This equation tells how to determine $H(f)$ from the coefficients a_m and b_m . However, filter design requires the inverse problem: how to obtain a suitable set of coefficients based upon knowledge of a desired filter transfer function. A variety of techniques exist to solve this problem, and the interested reader is referred to Stearns (1975) for a lengthy discussion of the matter. The transfer function discussed above is more commonly expressed in terms of the variable z where

$$z \equiv e^{i2\pi f\Delta t} \tag{3.12}$$

For recursive filters, $H(z)$ is a rational function of z. It seems that rational functions are particular good at fitting functions with sharp edges; most filter functions are in this category. The filters contained in this routine are modified versions of FORTRAN subroutines which originally appeared in Stearns (1975). The order of the filter is determined by the number of cascaded stages that are used; a filter of order P will require $P/2$ cascaded stages. The transfer functions, $H_k(z)$, for the kth stage for each filter, and the recursive algorithms required to implement those transfer functions are given below.

$$H_k(z) = \frac{A_k(z^2+2z+1)}{z^2+B_k z+C_k} \qquad \text{lowpass} \tag{3.13}$$

$$y_n = A_k(x_n + 2x_{n-1} + x_{n-2}) - B_k y_{n-1} - C_k y_{n-2}$$

$$H_k(z) = \frac{A_k(z^2-2z+1)}{z^2+B_k z+C_k} \qquad \text{highpass} \tag{3.14}$$

$$y_n = A_k(x_n - 2x_{n-1} + x_{n-2}) - B_k y_{n-1} - C_k y_{n-2}$$

$$H_k(z) = \frac{A_k(z^4-2z^2+1)}{z^4+B_k z^3+C_k z^2+D_k z+E_k} \qquad \text{bandpass} \tag{3.15}$$

$$y_n = A_k(x_n - 2x_{n-2} + x_{n-4}) - B_k y_{n-1} - C_k y_{n-2} - D_k y_{n-3} - E_k y_{n-4}$$

$$H_k(z) = \frac{A_k(z^4+2Kz^3+(2+K^2)z^2+2Kz+1)}{z^4+B_k z^3+C_k z^2+D_k z+E_k} \qquad \text{bandstop} \tag{3.16}$$

$$y_n = A_k(x_n + 2Kx_{n-1} + (2+K^2)x_{n-2} + 2Kx_{n-3} + x_{n-4}) - B_k y_{n-1} - C_k y_{n-2} - D_k y_{n-3} - E_k y_{n-4}$$

Note that the quantity K in equation 3.16 above is a constant which is the same for all stages of the notch filter. The other coefficients, A_k through E_k, are a more compact notation for the a_m and b_m values for each stage k. In general, A_k through E_k change from stage to stage and are evaluated from complicated expressions which the interested reader may find in the source code.

To use these filters, the user selects the desired filter type and enters the critical (-3 dB) frequency for the low and highpass filters. For the bandpass and bandstop filters, two critical frequencies must be specified. The number of filter sections to cascade must also be specified. The order of the filter is twice the number of cascaded sections. As the order of the filter is increased, the transfer function for the filter becomes sharper and more closely approximates the ideal; however, the amount of computation time is also significantly increased. A good starting point may be to use five cascaded sections (tenth order); this is a reasonable trade-off between accuracy and computational efficiency. These filters work well when the input data is not deterministic; however, unlike nonrecursive filters, recursive filters can become unstable and produce an output which grows exponentially. This has been observed in a few rare instances during testing of this program and, so far, has been found to occur only when the input data has been a *perfect* computer-generated sinusoid. The user should be aware of this limitation.

3.5 Tapering

The tapering option has been added to the routine as a *grooming* operation in order to improve the resulting spectral density estimates. A wide variety of window functions may be found in the literature including extended discussions concerning their performance characteristics. See, for example, Bendat and Piersol (1986) and Press, *et al.* (1986). Rather than repeat these long discussions here, a brief summary of the need for tapering will be followed by a description of the window functions available for use with this routine.

As discussed earlier in regard to equation 1.1, the data processed by this routine is in the form of a finite set of samples x_n of the continuous signal $x(t)$ which exists for a time $T = N\,\Delta t$. Now, suppose that $x(t)$ is a finite portion of a continuous signal $v(t)$ which is of unlimited extent in time. Then, $x(t)$ can be considered to be the product of $v(t)$ and a rectangular window function $w(t)$

$$x(t) \;=\; w(t)\, v(t) \tag{3.17}$$

where

$$w(t) \;=\; \begin{cases} 1 & 0 \le t \le T \\ 0 & \text{otherwise} \end{cases} \tag{3.18}$$

A plot of the rectangular (square) window function $w(t)$ is shown in figure 3.1 where, for convenience, T is chosen to equal one. The computation of spectral density estimates for $x(t)$ involves the calculation of the Fourier transform of $x(t)$ which is equivalent to the Fourier transform of the product of $w(t)$ and $v(t)$. A theorem from Fourier analysis relates the Fourier transform of a product of two functions in time to the *convolution* of the transforms of these functions. In other words, the following is a Fourier transform pair:

$$w(t)\, v(t) \;\Leftrightarrow\; W(f) * V(f) \tag{3.19}$$

where

$$W(f) * V(f) \;=\; \int_{-\infty}^{\infty} W(\alpha)\, V(f-\alpha)\, d\alpha \tag{3.20}$$

The spectral density of interest would be exact if the transform $V(f)$ could be calculated. Instead, $X(f) = W(f) * V(f)$ is the quantity which is, in fact, calculated. This problem then is seen to be directly related to the finite length of the data to be transformed and to the fact that the data is sharply truncated at each end of the time record. A plot of the modulus of the Fourier transform of the rectangular window function, $W(f)$, may be found in figure 3.2 and is defined as

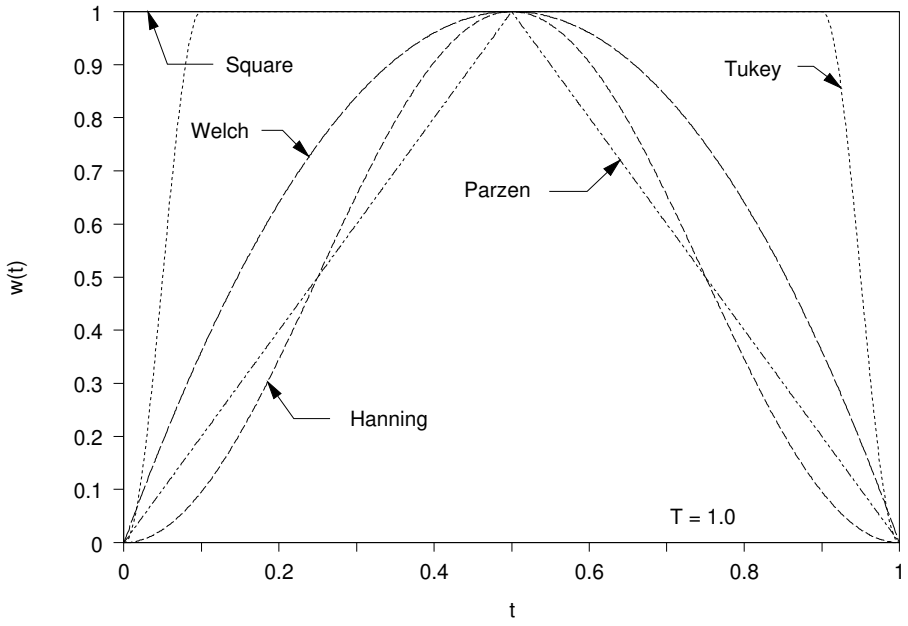

Figure 3.1 Window functions available for tapering the data

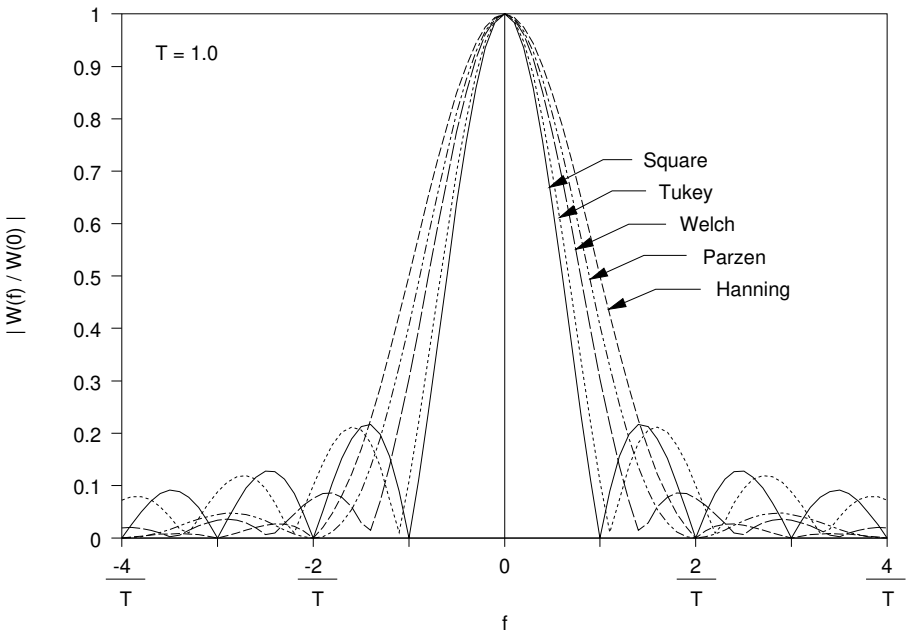

Figure 3.2 Normalized amplitude of Fourier transform of window functions

$$W(f) = \frac{\sin \pi f T}{\pi f} \; e^{-i\pi f T} \tag{3.21}$$

Again, for plotting convenience, T is chosen to equal one. This function is characterized by a large main lobe with smaller side lobes which slowly decrease in size. The ideal window function would be a delta function; that is, an infinitely thin main lobe with no side lobes at all. As $W(f)$ is convolved with $V(f)$, information at frequencies well separated from the main lobe can *leak* into the calculation and significantly distort the resulting spectra. The large amplitude sidelobes need to be reduced to suppress this problem.

This is accomplished by multiplying the original data by a window function which tapers the time history at the beginning and end of the record to remove the discontinuities there. The tapering functions included in this routine are the most commonly used ones, and the choice of a particular function requires a trade-off between the thickness of the main lobe (resolving power) and the size of the side lobes. The following window functions (the subscripts merely indicate the name of the window) may be selected:

$$w_h(t) = \begin{cases} \frac{1}{2}\left(1 - \cos\frac{2\pi t}{T}\right) & 0 \le t \le T \\ 0 & \text{otherwise} \end{cases} \qquad \text{Hanning} \tag{3.22}$$

$$w_t(t) = \begin{cases} \frac{1}{2}\left(1 - \cos\frac{10\pi t}{T}\right) & 0 \le t \le \frac{T}{10} \\ 1 & \frac{T}{10} \le t \le \frac{9T}{10} \\ \frac{1}{2}\left(1 - \cos\frac{10\pi t}{T}\right) & \frac{9T}{10} \le t \le T \\ 0 & \text{otherwise} \end{cases} \qquad \text{Tukey} \tag{3.23}$$

$$w_w(t) = \begin{cases} 1 - \left(\frac{t - \frac{T}{2}}{\frac{T}{2}}\right)^2 & 0 \le t \le T \\ 0 & \text{otherwise} \end{cases} \qquad \text{Welch} \tag{3.24}$$

$$w_p(t) = \begin{cases} 1 - \left| \dfrac{t - \frac{T}{2}}{\frac{T}{2}} \right| & 0 \le t \le T \\ 0 & \text{otherwise} \end{cases} \qquad \text{Parzen} \qquad (3.25)$$

These functions are plotted in figure 3.1. The moduli of the Fourier transforms for each of the above windows are plotted in figure 3.2 and are normalized by their value at $f = 0$. The fact that these transforms do not equal one for $f = 0$ implies that the resulting spectral estimates would be reduced in magnitude upon application of one of these windows. To avoid this problem, the appropriate scale factor is computed by the routine when any of these windows are selected and automatically applied to the data before computing the spectral estimate in order to preserve the correct amplitudes. The Fourier transform for each of the above windows is given below.

$$W_h(f,T) = \left\{ \frac{1}{2}\left(\frac{\sin \pi fT}{\pi f}\right) + \frac{1}{4}\left(\frac{\sin \pi(f-f_1)T}{\pi(f-f_1)}\right) + \frac{1}{4}\left(\frac{\sin \pi(f+f_1)T}{\pi(f+f_1)}\right) \right\} e^{-i\pi fT} \qquad (3.26)$$

$$\begin{aligned} W_t(f,T) &= \left\{ \frac{1}{2}\left(\frac{\sin \pi fT}{\pi f}\right) + \frac{1}{4}\left(\frac{\sin \pi(f-5f_1)T}{\pi(f-5f_1)}\right) + \frac{1}{4}\left(\frac{\sin \pi(f+5f_1)T}{\pi(f+5f_1)}\right) \right\} e^{-i\pi fT} \\ &- \left\{ \frac{1}{2}\left(\frac{\sin \pi f\frac{T}{10}}{\pi f}\right) + \frac{i}{4}\left(\frac{\sin \pi(f-5f_1)\frac{T}{10}}{\pi(f-5f_1)}\right) - \frac{i}{4}\left(\frac{\sin \pi(f+5f_1)\frac{T}{10}}{\pi(f+5f_1)}\right) \right\} e^{-i\pi f\frac{T}{10}} \\ &+ \left\{ \frac{1}{2}\left(\frac{\sin \pi f\frac{9T}{10}}{\pi f}\right) + \frac{i}{4}\left(\frac{\sin \pi(f-5f_1)\frac{9T}{10}}{\pi(f-5f_1)}\right) - \frac{i}{4}\left(\frac{\sin \pi(f+5f_1)\frac{9T}{10}}{\pi(f+5f_1)}\right) \right\} e^{-i\pi f\frac{9T}{10}} \end{aligned} \qquad (3.27)$$

$$W_w(f,T) = \frac{2}{\pi fT}\left(\frac{1}{\pi fT}\,\frac{\sin \pi fT}{\pi f} - \frac{\cos \pi fT}{\pi f}\right) e^{-i\pi fT} \qquad (3.28)$$

$$W_p(f,T) = \frac{1}{\pi fT}\left(\frac{1}{\pi f} - \frac{\cos \pi fT}{\pi f}\right) e^{-i\pi fT} \qquad (3.29)$$

In equations 3.26 and 3.27 above $f_1 = 1/T$. Recall that if none of these windows are selected (no tapering), then the square window defined in equation 3.18 and with a Fourier transform as defined in equation 3.21 is implicitly used. The most commonly used window function is the Hanning window; the transform for this function has the smallest side lobes but also has the widest main lobe. Also shown is a window which provides a cosine taper for the first and last tenths of the data record, is one for the remainder of the data record and is denoted by the name Tukey (also known as a 10% cosine taper). This window has been used by many researchers in the hopes that it will *throw away less of the data*. Examination of the modulus of the Fourier transform for this window shows that it manages to narrow the main lobe a small amount relative to the other windows, but it does this at the expense of large side lobes that are nearly as large as that of the square window (no tapering at all).

A better way to *throw away less of the data* is to use any of the windows available above and then to use overlapped processing techniques. Before describing overlapped processing, it is useful to review the procedure for the calculation of spectra when no overlap is used. The data are prepared (detrended, filtered, etc.), and then the data are tapered. The data are tapered record by record and transformed to get the necessary spectra. When 50% overlapped processing is used, the tapering operation proceeds a bit differently. First, the initial record of the data is tapered. Then, the second half of the first record is grouped together with the first half of the second record to form a new second record, and then this new record of data is tapered. Then, the two halves of the old second record form the new third record, and this is then tapered. Then, the second half of old second record is grouped together with the first half of the old third record to form a new fourth record which is then tapered. This proceeds throughout the file until M old records are processed creating exactly $2M$ new records. The last grouping pairs the second half of the old last record with the first half of the old first record to form the new $2Mth$ record. This last record was included in order to simplify the code for this procedure. Because the $2Mth$ record has a discontinuity at the midpoint, this record should not be transformed. However, simply removing this record would require that an *odd* number of records be submitted to the FFT code. As pointed out earlier, the FFT routine has been designed such that it transforms two records of data in one pass as a time-saving measure; this requires (in the one channel case) that an *even* number of records be processed by the FFT routine. Therefore, the overlapping operation has not been modified; $2M$ records are created from M original records for both the one and two channel cases. These $2M$ records are then submitted to the FFT subroutine and are transformed; but the results of the $2Mth$ record are discarded after the transformation. Summarizing, when overlapping is employed, $2M$ records are created and are then transformed, but only the results from $2M - 1$ records are actually used in the final calculations. To avoid discontinuities resulting from the use of overlapping in any of the other $2M - 1$ records, the M original records must be contiguous in time. The data need not be contiguous from record to record if overlapping is not employed. The use of 50% overlapped processing doubles the number of tapering and FFT operations; however, this will retrieve about 90% of the stability lost due to the tapering operation and will decrease the variability of the resulting estimate.

Finally, tapering is best used when the input data are essentially random data -- that is, not deterministic. When this program was tested, input data (both deterministic and random) with known spectral densities were used. When a sinewave was used for input, the power and amplitude spectra exhibited a lower noise floor when no tapering was used. On the other hand, when colored noise (bandwidth limited white noise) was used for input, a substantial improvement was evident in the output spectra when any of the four window functions described above was used to taper the data.

4 Spectrum Computation

This section introduces the definitions of the discrete Fourier transform and the various spectral density functions computed by this routine. The means by which these estimates are computed as well as the errors associated with the calculation are discussed. The need for additional code to adjust the value of the computed phase angle of the cross-spectral density function is explained by means of an example, and an explanation of the coherence function calculated by this routine is given.

4.1 Fast Fourier Transform

Using the notation of Bendat and Piersol (1986), the Fourier transform of a function $x(t)$, where $x(t)$ may be real or complex-valued, is given by

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi ft}\, dt \tag{4.1}$$

This definition is not particularly useful for experimentally derived data which exist for only a finite time interval. For this reason, one usually employs a finite-range transform over the interval $0 \le t \le T$, and this transform is defined by

$$X(f,T) = \int_{0}^{T} x(t)\, e^{-i2\pi ft}\, dt \tag{4.2}$$

As in equation 1.1, assume that each record of the input data file consists of N samples of $x(t)$ which have been obtained at equally spaced points in time with spacing Δt. Note that t_0 has been set equal to zero for convenience. Then, each record contains the values x_n where

$$x_n = x(n\Delta t) \qquad \text{for} \quad n = 0, 1, 2, \ldots, N-1 \tag{4.3}$$

Now, the finite-range transform defined in equation 4.2 can be written in discrete form as

$$X(f,T) = \Delta t \sum_{n=0}^{N-1} x_n\ e^{-i2\pi fn\Delta t} \tag{4.4}$$

Since only N input values of x_n are used in the computation of the transform, only N output values of $X(f,T)$ may be specified. These are typically chosen at the following discrete frequency values

$$f_k = \frac{k}{T} = \frac{k}{N\Delta t} = k\,\Delta f \qquad \text{for} \quad k = 0, 1, 2, \ldots, N-1 \tag{4.5}$$

Thus, the discrete form of the finite-range Fourier transform becomes

$$X(f_k, T) = \Delta t \sum_{n=0}^{N-1} x_n \; e^{-i\frac{2\pi k n}{N}} = \Delta t \; X_k \qquad \text{for} \quad k = 0, 1, 2, \ldots, N-1 \tag{4.6}$$

Although N output values have been specified, results are unique only out to $k = N/2$; this is a result of the symmetry of the Fourier transform for real input values. Thus, the highest frequency that can be resolved for a sampling rate of $1/\Delta t$ samples per second is the *Nyquist frequency* given by

$$f_{\frac{N}{2}} = f_c = \frac{1}{2\,\Delta t} \tag{4.7}$$

which occurs for $k = N/2$. To compute the X_k values in equation 4.6 requires N^2 *complex* multiply-add operations for each record of the data file; note that 1 complex multiply-add operation is equivalent to 4 real multiply-adds. This can be quite a formidable computational task as N becomes large. Fortunately, several algorithms for reducing the complexity and increasing the speed of the computation exist and are known collectively as *fast Fourier transforms* (FFT). Rather than discuss these algorithms in detail, the reader is referred to the discussions that may be found in the references. The FFT algorithm used here employs a technique known as time decomposition with input bit reversal. The kernel of this algorithm was taken from Stearns (1975). The simplest implementation of this algorithm calls for N to be a *power of 2*. This is required in this routine, and if necessary, each record of the input data file should be padded with zeros at the end to satisfy this requirement.

The FFT algorithm used in this routine is designed to handle the general case where $x(t)$ is a complex-valued function; that is, the input data which is passed to the FFT subroutine consists of two arrays: one containing the real parts and the other containing the imaginary parts. However, experimentally sampled random time series data are real-valued. When the FFT of a purely real-valued function is computed, certain symmetries in the resulting transform allow one to compute the FFTs for two real-valued functions simultaneously by inserting one into the input array of real parts and the other into the array of imaginary parts. The trick is to unscramble the individual transforms from the resulting computation. This speed improvement has been incorporated into this routine, but it places a restriction on the input data files. If the input data file to be processed by SPECTRUM consists of one channel of sampled data, then the data file must consist of an *even* number of records. The reason for this is that input data records are submitted to the FFT algorithm two at a time to obtain

the speed improvement discussed above. If the input data file consists of two channels of sampled data, then each record of the file is first decomposed into the two separate channels, and these are then submitted to the FFT algorithm to be simultaneously processed. So, for the case of two channels of sampled data, the input data file may have an even or odd number of records.

4.2 Autospectral (Power) Density

The one-sided autospectral density function, defined for positive frequencies only, is usually expressed as twice the amplitude of the Fourier transform of the autocorrelation function as follows:

$$G_{xx}(f) = 2 \int_{-\infty}^{\infty} R_{xx}(\tau)\, e^{-i2\pi f\tau}\, d\tau \tag{4.8}$$

$$\text{for } 0 \le f \le \infty, \text{ otherwise zero .}$$

where $R_{xx}(\tau)$ is the autocorrelation function defined as

$$R_{xx}(\tau) = E[\, x(t)\, x(t+\tau)\,] \tag{4.9}$$

and the notation E[ ] denotes the expected value operation. $G_{xx}(f)$ is a real function of f regardless of whether $x(t)$ is real or complex. This calculation is implemented in this routine using digital computing techniques employing finite fast Fourier transforms of the original data records. Using this method, the autospectral density may be obtained from

$$G_{xx}(f) = 2 \lim_{T \to \infty} \frac{1}{T} E[\, |X(f,T)|^2\,] \ . \tag{4.10}$$

| | denotes the modulus of a complex quantity, and $X(f,T)$ is the finite Fourier transform which is valid for the interval $0 \le t \le T$ and is calculated as in equation 4.6. If the input data file contains two channels of data, denoted as $x(t)$ and $y(t)$, then selection of the power spectral density option results in the computation of $G_{xx}(f)$ and $G_{yy}(f)$.

Specifically, the calculation proceeds by computing the FFT for each record of the input data file, determining the modulus of the complex result, squaring it and then saving the results to an array. This is repeated for each record; upon completion, the results are ensemble averaged to approximate the expected value operation required in the definition. The computation of an expected value is needed to reduce the random error of the estimated autospectral density function. The number of records in the input data file determines the

number of members in the ensemble which will be averaged. If the number of records in the data file (before tapering) is denoted by n_r, then the normalized random error of the autospectral density estimate is given by

$$G_{xx}(f): \qquad \varepsilon = \frac{1}{\sqrt{n_r}} \tag{4.11}$$

where the normalized random error is defined as a fractional quantity by dividing the standard deviation of the estimate by the actual value estimated. If the tapering operation is used, it will increase the variability of the resulting estimate by an amount which depends upon the particular window function chosen. This increase in the variance of the estimate is, in the worst case (Hanning), about two; but if the 50% overlap technique is also employed, it will counteract this increase and will recover about 90% of the variability caused by tapering. Thus, the normalized random error calculated from equation 4.11 will serve as a reasonably accurate estimate for both the no taper and taper with overlap cases. If tapering is used but overlapping is not, then the variance is doubled and the normalized random error defined in equation 4.11 should be multiplied by the square root of two. The routine computes the appropriate value of the normalized random error and displays it on the screen at run-time. As an example, consider an autospectral density estimate computed from a data file with 100 original records which are then overlapped and tapered to produce 199 records; this estimate will have a normalized random error of 10%. To reduce this error to 5%, the input data file would have to contain 400 records. There is no specific numerical limit to the number of records that an input data file may contain; the spectrum routine has successfully computed spectral densities for input data files with 10,000 records. However, data files with more than 100 records are usually quite large. Furthermore, the spectrum routine must create temporary files during computation, and the size of these files (depends upon the size of the input data file) is often very large. Reserving a sufficient amount of space on the hard disk or on a RAM disk for these files is typically the limiting factor on the number of records.

The computation of a fast Fourier transform and power spectral density function allows the calculation of the mean and root-mean-square (rms) values of each record of the input time series. The equation for the mean value of each data record can be obtained by setting $k = 0$ in equation 4.6. One then obtains

$$\bar{x} = \frac{X_0}{N} = \frac{1}{N} \sum_{n=0}^{N-1} x_n \tag{4.12}$$

The mean-square value of each record can be computed from

$$R_{xx}(0) = E[x^2(t)] = \int_0^\infty G_{xx}(f)\, df \tag{4.13}$$

The rms value is just the square root of this number. The spectrum routine calculates the mean and rms values of the input time series using this procedure and reports them on the screen at run-time along with the appropriate value of the normalized random error.

4.3 Cross-spectral Density

4.3.1 Calculation Procedure

A one-sided cross-spectral density function $G_{xy}(f)$ may be defined in a manner similar to equation 4.8; this quantity is, in general, complex and is given by

$$G_{xy}(f) = 2\int_{-\infty}^{\infty} R_{xy}(\tau)\, e^{-i2\pi f\tau}\, d\tau = C_{xy}(f) - i\, Q_{xy}(f) \qquad (4.14)$$

$$\text{for } 0 \le f \le \infty, \text{ otherwise zero .}$$

The quantity $R_{xy}(\tau)$ is the cross correlation function defined by

$$R_{xy}(\tau) = E[\, x(t)\, y(t+\tau)\,] \qquad (4.15)$$

The cross-spectral density may also be computed using fast Fourier transform procedures and is the method adopted here. The equivalent expression to equation 4.10 for the one-sided cross-spectral density function is

$$G_{xy}(f) = 2 \lim_{T\to\infty} \frac{1}{T} E[\, X^{*}(f,T)\; Y(f,T)\,] \qquad (4.16)$$

where the notation $X^{*}(f,T)$ is used to represent the complex conjugate. Alternatively, this can be expressed in polar notation as

$$G_{xy}(f) = |\, G_{xy}(f)\,|\, e^{-i\theta_{xy}(f)} \qquad 0 \le f < \infty$$

$$\text{where} \qquad |\, G_{xy}(f)\,| = [C_{xy}^{2}(f) + Q_{xy}^{2}(f)]^{\frac{1}{2}} \qquad (4.17)$$

$$\text{and} \qquad \theta_{xy}(f) = \tan^{-1}\frac{Q_{xy}(f)}{C_{xy}(f)} \; .$$

Note that this option may be selected only when the input data file contains two channels of data, then, the magnitude and phase, $|G_{xy}(f)|$ and $\theta_{xy}(f)$ respectively, are the quantities which are calculated and then saved in the output file.

As described above in the section on power spectral density calculation, it is possible to compute the value of the autocorrelation function for $\tau = 0$ (the mean-square value) by calculating the integral of the autospectral density function. In a similar manner one may compute the cross correlation function evaluated at $\tau = 0$ from

$$R_{xy}(0) = E[x(t)\, y(t)] = \int_0^\infty C_{xy}(f)\, df \tag{4.18}$$

The spectrum routine calculates this quantity and displays it on the screen at run-time.

4.3.2 Adjustment of Phase Angle

In order to clarify the need for additional code to adjust the values of the phase angle $\theta_{xy}(f)$, an example taken from Hess (1990) will be given which also serves to illustrate the use of the cross spectral density function in general. Specifically, a point measurement of the vertical displacement of a compliant surface, responding to the unsteady forcing of a fluid flow above the surface, was acquired; call this displacement signal $x(t)$. As $x(t)$ was measured, a second displacement signal was simultaneously measured; call this second signal $y(t)$. The locations on the compliant surface of the measurement of $x(t)$ and $y(t)$ were as follows: both were obtained on the centerline of the rectangular compliant slab, and $y(t)$ was located 11.4 cm downstream of $x(t)$. The fluid flow was turbulent; the instantaneous streamwise component of velocity above the compliant surface consisted of a small fluctuating quantity superimposed upon a larger mean value. A disturbance in the fluid flow caused a displacement of the compliant surface which was measured at the upstream location and was recorded at time t as $x(t)$. As the flow disturbance was convected downstream by the mean flow, the displacement of the compliant surface also travelled downstream and was measured at the downstream location as $y(t+\tau)$ where τ was the time delay for the disturbance to travel 11.4 cm. Quantities of interest, then, are the convection speed of the displacement fluctuation on the compliant surface as well as the decrease in amplitude of this fluctuation as it decays.

For this particular situation, the decay of the displacement field may be modeled by the time delay problem, see Bendat and Piersol (1986). This model provides a method for determining the convection speed of the displacement fluctuation. Consider two zero-mean stationary random signals $x(t)$ and $y(t)$ where $y(t)$ is defined by

$$y(t) = \alpha x(t-\tau_0) + n(t)\ . \tag{4.19}$$

The value α is a constant attenuation, τ_0 is a constant time delay and $n(t)$ is uncorrelated, zero mean value noise at the output. This process is illustrated in figure 4.1.

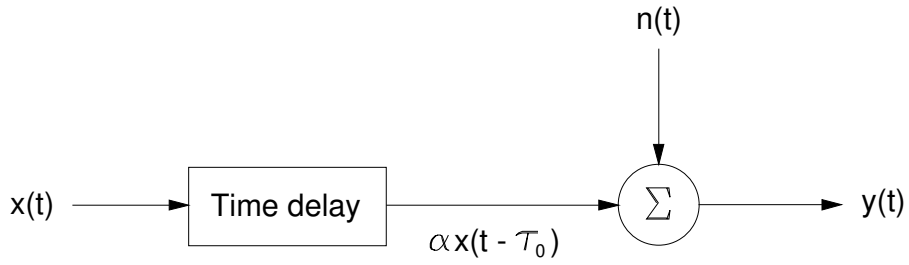

Figure 4.1 Schematic for time delay problem

The cross-correlation of these two signals can be computed to be

$$R_{xy}(\tau) = \alpha R_{xx}(\tau - \tau_0) \ . \tag{4.20}$$

Thus, the cross-correlation is an attenuated replica of the autocorrelation, which is displaced in time by the amount τ_0. Without going through all of the details, a one-sided cross-spectral density function $G_{xy}(f)$ may also be computed; this quantity is, in general, complex and for this problem is given by

$$G_{xy}(f) = \alpha G_{xx}(f) e^{-i2\pi f\tau_0} \tag{4.21}$$

$$\text{with } |G_{xy}(f)| = \alpha G_{xx}(f) \text{ and } \theta_{xy}(f) = 2\pi\tau_0 f \ .$$

Since the time delay τ_0 appears only in the phase angle, $\theta_{xy}(f)$, it can be determined from measurement of the phase angle and by noting that it is a linear function of the frequency with slope $2\pi\tau_0$. Summarizing then, a cross-spectral density is computed for $x(t)$ and $y(t)$. If the process can be modelled by the time delay problem, then the modulus will be an attenuated replica of the autospectral density, $G_{xx}(f)$, and the phase angle will be a linear function of the frequency with a slope given by $2\pi\tau_0$. A convection speed may then be calculated from $U_c = \frac{11.4}{\tau_0}$ cm/sec.

An example of this procedure is shown in figure 4.2. The top graph shows the auto-spectral density curve $G_{xx}(f)$ and below it the modulus $|G_{xy}(f)|$ for the corresponding cross-correlation; indeed, the bottom curve does appear to be an attenuated copy of the first. Note the substantial additional scatter in the cross-spectral density estimate relative to that for the autospectral density estimate at the higher frequencies. The normalized random error

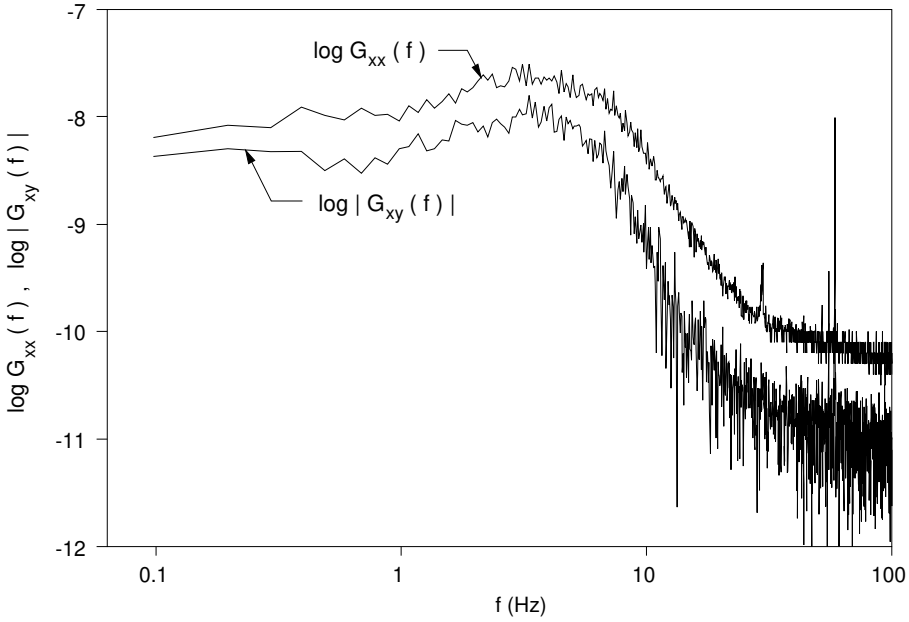

Figure 4.2 Cross-spectral density modulus example

for the cross-spectral density estimate turns out to be a function of frequency unlike that for the autospectral density estimate. A discussion of the means for estimating this error is given in a later section.

Figure 4.3 displays the computed phase angle expressed in radians. With the data presented in this form, it is difficult to fit the data with a least squares fit in order to determine the slope, and therefore, τ_0 . This difficulty may be traced to the fact that an inverse tangent must be computed in order to calculate the value of the phase angle. The FORTRAN intrinsic function ATAN2 returns a value for the inverse tangent, but forces the result to lie in the range $-\pi \le result \le \pi$. However, the phase angle function is periodic with period 2π . Using this fact, additional code may be used to *straighten out* the curve by adding integer multiples of 2π as required. Figure 4.4 shows the adjusted values of the phase angle and a least squares fit to the slope. The calculated values of τ_0 and U_c / U_∞ , where U_∞ is the mean flow speed far above the compliant surface, are given on the plot. The scatter in the phase angle data becomes large at a frequency for which the energy in the signal is about two decades below the peak value. Computed values of the convection speed using this method vary by less than 2% from those obtained using an alternative method.

The phase angle data can be adjusted manually, but this is a tedious and time-consuming procedure. Instead, the spectrum routine automatically computes both the original values of the phase angle as well as the adjusted values of the phase angle when the cross spectral density and the coherence function are calculated. Note that the relationships derived in equations 4.21 apply only when equation 4.19 may be used as a model for the data. Therefore,

both sets of phase angle data, unadjusted and adjusted values, are stored to the output file and made available to the user. The task of adjusting these phase angle values, while conceptually simple, is somewhat tricky to program. A description of the technique follows.

The standard deviation of the phase angle estimate as a function of frequency is computed when the coherence function is computed (this will be explained in the section on error estimates). The average of a moving window of NSD values of this standard deviation is computed and checked against a threshold value given by $SDLIM$. If the average standard deviation is too high, then the phase angle points are considered to be essentially random and no further attempts to adjust the values are made. Thus, when this condition occurs, the remaining phase angle values are kept the same as the original values (not adjusted). When the moving average of the standard deviation is less than $SDLIM$, then adjustment may take place as follows. A straight line is fit to a moving window of $NPTS$ values of the original phase angle data using a least squares fit. The coefficients of this fit are used to guess the next value of the phase angle; if the actual value of the next phase angle data point differs from the guessed value by an amount greater than π radians, then multiples of 2π radians are added to (or subtracted from) the actual value until the difference between the actual and guessed values is less than π radians. The procedure continues to adjust subsequent values of the phase angle until the moving average of the standard deviation exceeds $SDLIM$. This portion of the code contains three adjustable parameters: $NSD = 7$, $NPTS = 7$ and $SDLIM = 70.0°$. These values have been selected by trial and error to yield the best response for the data most often processed by the author. Other values may be chosen by changing the numbers in the appropriate PARAMETER statement at the beginning of the code. Note that NSD and $NPTS$ must be chosen such that

$$NSD \leq 2\,NPTS - 1 \tag{4.22}$$

is satisfied. This adjustment procedure seems to work reasonably well for most situations encountered by the author; however, some manual adjustment of values may still be required in some circumstances.

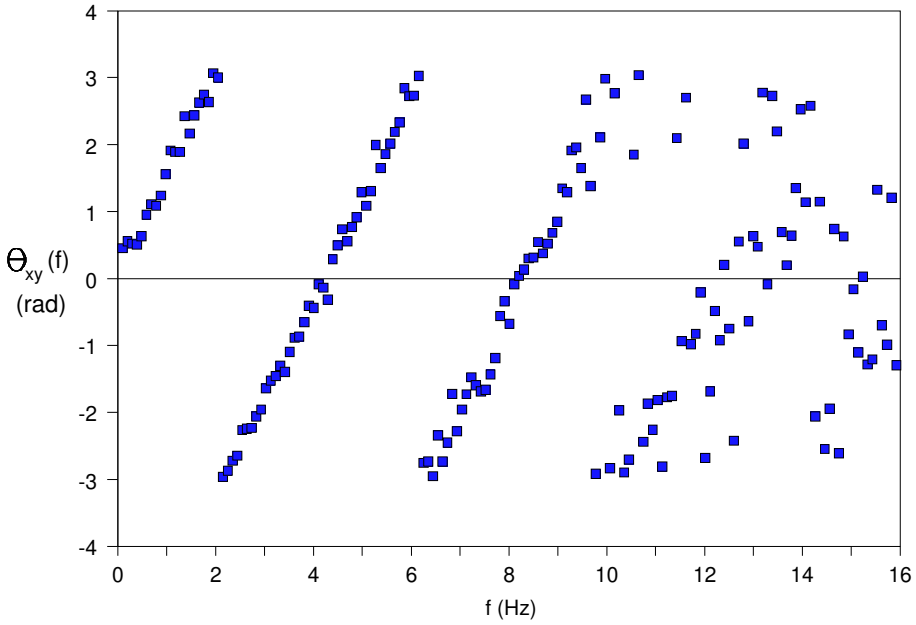

Figure 4.3 Phase angle example

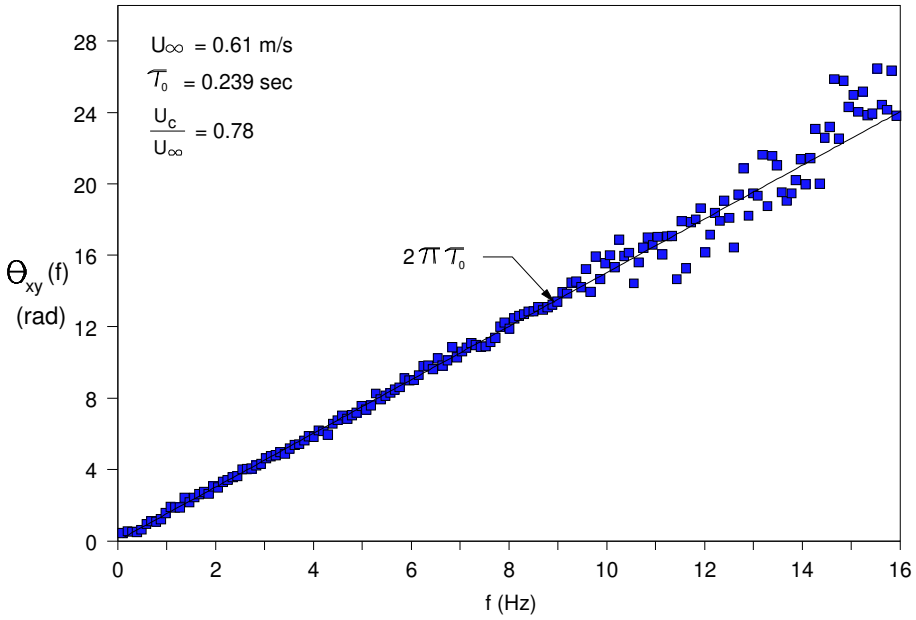

Figure 4.4 Phase angle - adjusted values

4.3.3 Coherence Function

The coherence function, also known as the *coherency squared function* is defined by

$$\gamma_{xy}^2(f) = \frac{|\, G_{xy}(f)\,|^2}{G_{xx}(f)\;\; G_{yy}(f)} \tag{4.23}$$

where $0 \le \gamma_{xy}^2(f) \le 1$ for all f. This function is analogous to the square of the more well known correlation coefficient function given by

$$\rho_{xy}(\tau) = \frac{R_{xy}(\tau)}{[R_{xx}(0)\;\; R_{yy}(0)]^{1/2}} = \frac{E[\, x(t)\; y(t+\tau)\,]}{[\, E[\, x^2(t)\,]\;\; E[\, y^2(t)\,]\,]^{1/2}} \tag{4.24}$$

where $-1 \le \rho_{xy}(\tau) \le 1$ for all values of τ. As described by Bendat and Piersol (1986), the function $\rho_{xy}(\tau)$ measures the degree of linear dependence between $x(t)$ and $y(t)$ for a displacement of τ in $y(t)$ relative to $x(t)$. Similarly, for linear systems, the function $\gamma_{xy}^2(f)$ measures the fractional portion of the mean square value at the output $y(t)$ that is contributed by the input $x(t)$ at frequency f. On the other hand, the quantity $[1 - \gamma_{xy}^2(f)]$ is a measure of the mean square value of $y(t)$ not accounted for by $x(t)$ at frequency f. If $x(t)$ and $y(t)$ are completely unrelated, the coherence function will be zero. Furthermore, Bendat and Piersol (1986) indicate that if the coherence function is greater than zero but less than unity, one or more of the following three possible physical situations exist.

1. Extraneous noise is present in the measurements.
2. The system relating $x(t)$ and $y(t)$ is not linear.
3. $y(t)$ is an output due to an input $x(t)$ as well as to other inputs.

The calculation of this function requires not only the computation of the cross spectral density function but also the computation of power spectral densities for each channel of data. The code has been optimized to minimize the amount of additional computational effort; however, in anticipation of the possibility that a slow computer might be employed, the routine prompts the user for this option at run-time.

4.3.4 Error Estimates

The definition of the coherence function makes possible at this point a discussion of the error associated with estimates of the magnitude and phase of the cross spectral density function and of the coherence function itself. This information comes from Bendat and Piersol (1986). The normalized random error of the estimate of the magnitude of the cross spectral density function is given by

$$| G_{xy}(f) | : \qquad \varepsilon(f) = \frac{1}{| \gamma_{xy}(f) | \sqrt{n_r}} \tag{4.25}$$

where $| \gamma_{xy}(f) |$ is the positive square root of the coherence function evaluated at f, and n_r is the number of records in the input data file. The error approaches that for the power spectral density estimate (see equation 4.11) as $| \gamma_{xy}(f) |$ goes to one, and the error is higher for values less than one. Unlike the error estimate for the power spectral density function, all of the error estimates in this section are a function of frequency.

Since the phase angle may assume the value zero, a normalized random error cannot be specified; instead the standard deviation (s.d.) of the phase angle estimate is given by

$$\theta_{xy}(f) : \qquad s.d.(f) = \frac{[1 - \gamma_{xy}^2(f)]^{1/2}}{| \gamma_{xy}(f) | \sqrt{2 n_r}} \tag{4.26}$$

The standard deviation is seen to approach zero as $| \gamma_{xy}(f) |$ goes to one and does so independent of n_r for $n_r > 1$. Continuing, the normalized random error of the coherence function itself can also be calculated and is defined as

$$\gamma_{xy}^2(f) : \qquad \varepsilon(f) = \frac{\sqrt{2} [1 - \gamma_{xy}^2(f)]}{| \gamma_{xy}(f) | \sqrt{n_r}} \tag{4.27}$$

where the estimated value of the coherence function is substituted as an approximation to the unknown true values of $\gamma_{xy}^2(f)$ and $| \gamma_{xy}(f) |$ called for in equation 4.27. Note that the normalized random error approaches zero as either $n_r \to \infty$ or $\gamma_{xy}^2 \to 1$. This implies that coherence function estimates can be more accurate than either autospectra or cross spectra estimates (which are used to calculate the coherence function) when $\gamma_{xy}^2(f)$ is close to unity. If the coherence function is calculated, the spectrum routine automatically computes the three error estimates defined in equations 4.25 to 4.27 as a function of frequency and saves the results in a separate output file. The file name and structure of this file will be discussed in the section on output files. Equations 4.25 to 4.27 are valid for the no taper and taper with overlap cases. If tapering is used without overlapping, then the error estimates in equations 4.25 to 4.27 must each be increased by multiplying by the square root of two, and this is automatically performed by the spectrum routine.

It is perhaps useful to know the number of records required in the input data file to attain a specified normalized random error or standard deviation for each of the estimates mentioned above. This has been calculated for various values of $\gamma_{xy}^2(f)$ for the cases: $\varepsilon [| G_{xy}(f) |] = 0.1$, $\varepsilon [\gamma_{xy}^2(f)] = 0.1$ and $s.d. [\theta_{xy}(f)] = 0.052\ rad \approx 3°$, and the results are given in table 4.1 below.

$\gamma_{xy}^2(f)$	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\varepsilon [\| G_{xy}(f) \|] = 0.1$	334	250	200	167	143	125	112
$s.d. [\theta_{xy}(f)] \approx 3°$	426	274	183	122	79	46	21
$\varepsilon [\gamma_{xy}^2(f)] = 0.1$	327	180	100	54	26	10	2

Table 4.1 Number of records for specified error

4.4 Amplitude Spectral Density

The amplitude spectrum option was added to allow the user to examine the Fourier coefficients directly. The amplitude spectrum is defined as

$$F_x(f) = \lim_{T \to \infty} E [| X(f,T) |] \tag{4.28}$$

where $| X(f,T) |$ is the modulus of the quantity calculated in equation 4.6. If the input data file contains two channels of data, denoted as $x(t)$ and $y(t)$, then selection of this option results in the computation of $F_x(f)$ and $F_y(f)$.

A simple example taken from Stearns (1975) is used to demonstrate the utility of this option. Consider the following function which was computer-generated and then saved as a set of digital values to simulate the sampling process:

$$x(t) = e^{-t} \sin t \tag{4.29}$$

A plot of a portion of this function is shown in figure 4.5. The Fourier transform of this function can be derived using the definition in equation 4.1 as

$$X(f) = \frac{1}{(i\ 2\pi f + 1)^2 + 1} \tag{4.30}$$

and the modulus of this quantity becomes

$$| X(f) | = \frac{1}{[(2\pi f)^4 + 4]^{1/2}} \tag{4.31}$$

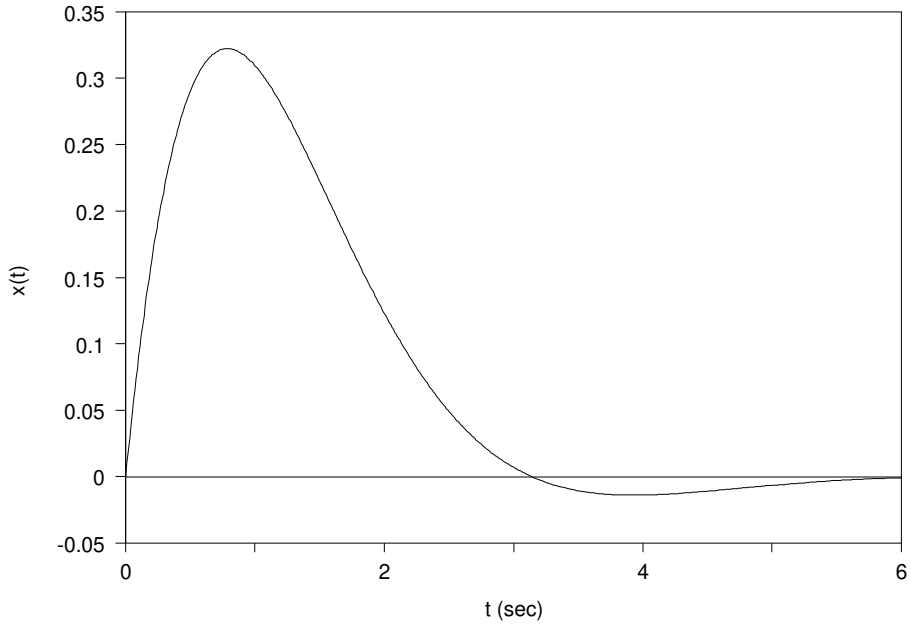

Figure 4.5 Plot of example input function

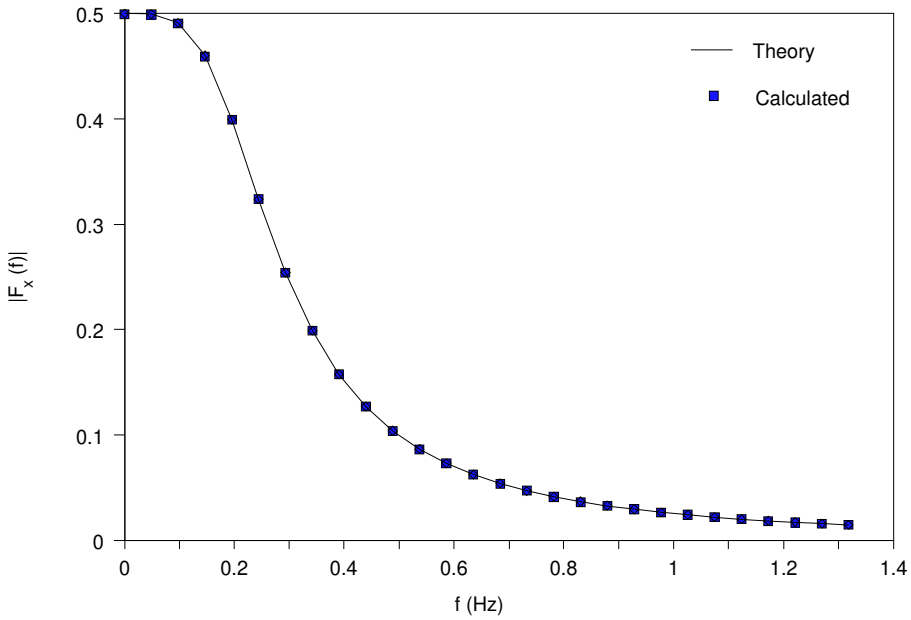

Figure 4.6 Plot of amplitude spectrum calculation

This input function is deterministic, and therefore only one record of data would normally be required; however, because an even number of records are required by the spectrum routine for one channel of input data, two identical records of this function were generated. $N = 2048$ samples per record were created with a time interval of $\Delta t = 0.01$ seconds between successive samples. A portion of the results is shown in figure 4.6 with the symbols representing the values computed by the spectrum routine and the continuous line the values generated by equation 4.31.

5 Output Data Files

For every input data file processed, the spectrum routine creates one (or sometimes two) ASCII output files containing the results. A second output file is created only if the user processes two channels of input data and computes the cross-spectral density and the coherence function; the second output file then contains error estimates of cross-spectral magnitude and phase and of the coherence function.

5.1 Naming Convention

The output file name will consist of nine characters. The last four characters specify the file extension; the default file extension, [.PRN], will be used unless specified otherwise in the configuration file. The first four characters of the output file name will match the first four characters of the corresponding input data file. The fifth character is used to designate the type of file that is being written. The four possibilities are:

power spectrum	P
cross spectrum	C
amplitude spectrum	A
error estimates (if computed)	E

The output file will be written to the same location as the input data file; this will, by default, be the location where the executable version of the spectrum routine resides. This location can be changed, however, in the configuration file.

5.2 File Structure

These output data files consist of ASCII data as opposed to binary data. They may be easily edited by any ASCII editor, imported into spreadsheet programs for further manipulation or imported into plotting programs to be graphed. These files contain numbers only; no labels are included to describe the various columns of data in the output. While somewhat inconvenient for the user, this was done to prevent the confusion that sometimes occurs when data containing both numbers and labels are imported into other software programs. In the sections that follow, a thorough description of the output to be expected for each of the four types of output files is given, and this should allow the user to interpret the results without any difficulty.

5.2.1 Power Spectrum Output Files

The first line of the output contains the mean and rms values computed as described in equations 4.12 and 4.13 and then ensemble-averaged over all records of the file. Note that the mean and rms values reported here are calculated subsequent to any cosmetic operations that may have been applied to the data to improve the quality of the spectral density computation. Removal of the mean value, detrending, filtering and tapering operations may modify the mean and rms so that they do not match the original values for the file which were calculated and reported at run-time prior to the application of these

operations. Continuing, the next line of the output file is blank. The calculated results begin on the third line and consist of four or six columns of data depending on whether the input file contains one or two channels. The quantities are stored in the following order:

One channel:

mean rms

skip a line

f	$\log f$	$G_{xx}(f)$	$\log G_{xx}(f)$
$\frac{1}{N\Delta t}$	…	…	…
.	.	.	.
.	.	.	.
.	.	.	.
$\frac{1}{2\Delta t}$	…	…	…

Two channels:

mean rms

skip a line

f	$\log f$	$G_{xx}(f)$	$\log G_{xx}(f)$	$G_{yy}(f)$	$\log G_{yy}(f)$
$\frac{1}{N\Delta t}$	…	…	…	…	…
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
$\frac{1}{2\Delta t}$	…	…	…	…	…

In either case the output file contains $N/2$ rows of data after the blank line with each row calculated for values of f ranging from $1/N\Delta t$ to $1/2\Delta t$ with a resolution of $\Delta f = 1/N\Delta t$. This structure is typical for all of the output files created by the spectrum routine, and in the following descriptions of the output only the column labels will be identified.

5.2.2 Cross Spectrum Output Files

A cross spectrum may only be calculated when the input file contains two channels of data. However, the output file is somewhat different depending upon whether or not the coherence function is calculated. In either case, the file begins with the value of the cross

correlation function evaluated at $\tau = 0$ and calculated according to equation 4.18. The next line is skipped, and the following record is the first of $N/2$ rows of data arranged in five or seven columns as described below.

Coherence function not calculated:

$R_{xy}(0)$

skip a line

$f \quad \log f \quad |G_{xy}(f)| \quad \log|G_{xy}(f)| \quad \theta_{xy}(f)$

Coherence function calculated:

$R_{xy}(0)$

skip a line

$f \quad \log f \quad |G_{xy}(f)| \quad \log|G_{xy}(f)| \quad \theta_{xy}(f) \quad \theta_{xy}^{adjusted}(f) \quad \gamma_{xy}^2(f)$

5.2.3 Amplitude Spectrum Output Files

When the amplitude spectrum option is chosen, the structure of the output file is slightly different than for the other files. No mean or rms value is reported. Instead, the file contains $N/2+1$ rows of data arranged in the columns shown below. The first row consists of data computed for $f = 0$; data for this value of f is not stored when other options are selected, but is useful when examining the Fourier coefficients. Actually, the value $f = \varepsilon$ is used; this is necessary since $\log(0)$ is undefined. The value of ε is defined in a PARAMETER statement at the beginning of the code and is currently set at $\varepsilon = 1.0\ x\ 10^{-20}$.

One channel:

f	$\log f$	$F_x(f)$
ε	…	…
$\frac{1}{N\Delta t}$	…	…
.	.	.
.	.	.
.	.	.
$\frac{1}{2\Delta t}$	…	…

Two channels:

f	$\log f$	$F_x(f)$	$F_y(f)$
ε	$\cdots$	$\cdots$	$\cdots$
$\frac{1}{N\Delta t}$	$\cdots$	$\cdots$	$\cdots$
.	.	.	.
.	.	.	.
.	.	.	.
$\frac{1}{2\Delta t}$	$\cdots$	$\cdots$	$\cdots$

5.2.4 Error Estimate Output Files

When both the cross-spectral density and the coherence function options are selected, the spectrum routine calculates the errors associated with these estimates as a function of frequency using equations 4.25 through 4.27. These error data are then stored in a separate file from the other results. This file contains $N/2$ rows of data arranged in the columns identified below. The columns containing normalized random errors consist of dimensionless numbers, and the standard deviation of the phase angle is expressed in degrees.

$$f \quad \log f \quad \varepsilon\,[\,|\,G_{xy}(f)\,|\,] \quad s.d.\,[\,\theta_{xy}(f)\,] \quad \varepsilon\,[\,\gamma^2_{xy}(f)\,]$$

6 Run-time Considerations

Two topics are discussed in this section which assume importance when the program is executed. The creation of a *configuration* file allows the user to bend some of the rules regarding file name creation and the location of data files as well as temporary files created by the spectrum routine. The second topic concerns the manner in which the size of the temporary files created by the routine may be determined.

6.1 Configuration File

The presence of an ASCII file with the name SPECTRUM.CFG in the same directory as the SPECTRUM.EXE file allows optional settings to take effect which may have a favorable impact on the execution speed and overall ease of use of the routine. This ASCII file allows the user to specify three parameters which will become default values when the program is executed.

1. The first item is the drive and path information for the location of input, output and coefficient data files. If the user wishes to specify that these files may be found in (and written to) the same location as the SPECTRUM.EXE file, then the word **ROOT** should be entered on a line by itself. Otherwise, the desired drive and path information should be entered in the following format: C:\SPECTRUM\DATA\
   Note that the last backslash is required.
2. The second item in the file is the drive and path information which specifies where temporary data files created by the spectrum routine will be stored. If a RAM disk is present on the user's computer and is large enough to hold the temporary files, then the execution speed of the routine will be greatly enhanced if these files are located on this RAM disk. A means for estimating the size of these temporary files is discussed below. Again, the user specifies **ROOT** or the desired drive and path information using a format such as: D:\TEMP\
3. The last item allows the user to change the file extension for output data files from its default value of [.PRN] to some other desired value. The user should place the extension on a line by itself in the following format: .OUT
   Note that the leading period is required.

Comment lines may be included in the configuration file; these lines must start with an asterisk character. The routine is not case-sensitive -- the options may be specified in upper or lower case. The spectrum routine does not require the presence of the configuration file; if this file is absent, then the following default values are used: 1. root 2. root 3. [.prn] If a configuration file is used, then it must reside in the same directory as the SPECTRUM.EXE file and all three options must be specified. An example configuration file follows.

```
*       This is a sample configuration file for Spectrum.  Comment lines
*       and blank lines must begin with an asterisk.
*
*       The first item in the file is the path where the input data
*       files are found and to which the output data will be written.
*       Acceptable values are: root, c:\ , g:\ , c:\test\bin\ , etc.  The
*       path must end with a backslash if root is not used.
root
*       The second item in the file is the drive on which the temporary
*       files are to be created.  Program execution speed will increase
*       dramatically if this is a RAM disk.
*       Acceptable values are: root, c:\ , g:\ , c:\test\bin\ , etc.  The
*       path must end with a backslash if root is not used.
d:\
*       The third item in the file is the default file extension to use
*       for the output data files.  If not specified in this file, then
*       the .PRN extension will be used in order that the data may be
*       imported into LOTUS for plotting.  The extension must consist of
*       a period followed by three letters.
.prn
```

6.2 Temporary Files

The input data files that are to be processed by the spectrum routine are typically very large; therefore, the data is analyzed one record at a time. Furthermore, the final result of the calculation is not arrived at in one pass; instead, intermediate results are generated and stored in temporary files which are then used to obtain the final results. These temporary files are automatically deleted upon completion; however, sufficient space must be reserved on the hard disk or on a RAM disk for temporary use by these files. The following discussion explains how these files are created and allows the user to estimate their size based upon the size of the input data file. Note that if several input files are to be processed by the routine, only the temporary files for the input data file currently being processed reside on the hard disk at any one time; the user needs to allow sufficient space based upon the size of the largest input data file.

Two temporary files are created for each channel of input data. After a record of the input data file is read in by the routine, the integer data is sorted into one or two arrays depending on the number of input channels; each channel is then transformed into floating point data (if necessary), the mean value or a linear trend is removed and any desired filtering takes place. These data are then stored to temporary files with the extensions : [.TRF] - channel 1 and [.TR2] - channel 2 (if required). This process is repeated until all records of the input data file are read. If the input data file consisted of 2-byte integers, then the [.TRF] file will be twice the size of the input [.DAT] file since the [.TRF] file contains 4-byte floating point numbers. If the input data file consists of two channels of integer data, then

the [.TRF] and [.TR2] files will each be the same size as the [.DAT] file (the [.TRF] and [.TR2] files each contain half as many numbers as the input [.DAT] file, but each number is a 4-byte value). If the input data file consists of one channel of 4-byte real numbers, the [.TRF] file will be the same size as the [.DAT] file, and finally, for two channels of 4-byte values in the input file, the [.TRF] and [.TR2] files will each be half the size of the [.DAT] file. After creating the [.TRF] and [.TR2] (if required) files, the spectrum routine proceeds to the tapering operation. If tapering is selected, the routine reads a record of data at a time from the [.TRF] and [.TR2] files, creates two new records for each channel (if overlapping is used) and stores the results in two temporary files with the extensions [.TAP] and [.TP2]. The last step in the process is to transform the data stored in the [.TAP] and [.TP2] files and perform the required spectral calculation. The final data is written to one output file regardless of the number of channels of input data and the extension [.PRN], or alternatively an extension specified in the configuration file is used. If tapering is not selected, then the [.TAP] and [.TP2] files are not created, and storage space does not need to be allocated. If the [.TAP] and [.TP2] files are created, each will be twice the size of the previous [.TRF] or [.TR2] file, since two tapered records are written for every input record read to implement the 50% overlap technique. (If tapering is employed but overlapping is not, then the [.TAP] and [.TP2] files will be the same size as the [.TRF] or [.TR2] file. The table belows gives the worst case scenario and assumes overlapping is used when tapering is used.) At the end of the calculation, only the original [.DAT] file and the final [.PRN] file will remain. If no alternative is specified in the configuration file, the temporary files will be created in the same directory as the input data files. Several examples follow which show the sizes of the various files based upon an input data file size of 1 Mb.

Two Channels ? Integers ? Tapering ?	No Yes Yes	Yes Yes Yes	No No Yes	Yes No Yes	No Yes No
[.DAT]	1 Mb	1 Mb	1 Mb	1 Mb	1 Mb
[.TRF]	2 Mb	1 Mb	1 Mb	500 kb	2 Mb
[.TR2]	--	1 Mb	--	500 kb	--
[.TAP]	4 Mb	2 Mb	2 Mb	1 Mb	--
[.TP2]	--	2 Mb	--	1 Mb	--
[.PRN]	64 kb	93 kb	48 kb	83 kb	64 kb
Total Temp space needed	6 Mb	6 Mb	3 Mb	3 Mb	2 Mb

Table 4.2 Memory required for temporary files

7 Examples

Three simple examples are considered that serve to illustrate various features of the spectrum routine. The example using colored noise demonstrates the use of tapering and allows the calculated power spectrum to be compared with a known power spectrum. The computation of the amplitude spectrum of a sinewave reveals the importance that sampling parameters can have on the computed result, and the last example shows the effects of the application of a variety of digital filters prior to the computation of power spectra. A few comments regarding the considerations necessary for the acquisition of experimental data files are given at the end.

7.1 Colored Noise

A routine to produce bandwidth-limited white noise (colored noise) has been written and used extensively during the testing phase of the development of the spectrum routine. This program prompts the user for a few parameters, then generates the colored noise sequence and stores the output data in a manner that simulates the sampling process. This data may then be input into the spectrum program, and the output power spectrum estimate may be compared with the known power spectrum for colored noise. This handy routine is a useful utility program and has been included on the diskette with the name COLOR.

The procedure for producing colored noise begins by generating random numbers that vary from 0 to 1. This sequence is uniformly distributed over the interval from 0 to 1. Let x_n be the nth random number, then $x_n - 1/2$ will be uniformly distributed over the interval from $-1/2$ to $1/2$. Now, recall that the mean and variance of a uniformly distributed random variable is given by

$$\mu_x = \frac{a+b}{2} \qquad \text{and} \qquad \sigma_x^2 = \frac{(b-a)^2}{12} \tag{7.1}$$

where a and b are the endpoints of the interval. Using these definitions, we find that the mean and variance of the sequence $x_n - 1/2$ is

$$\mu_x = 0 \qquad \text{and} \qquad \sigma_x^2 = \frac{1}{12} \tag{7.2}$$

If a power spectrum for this sequence were calculated, then one would obtain the mean value, μ_x, the mean-square value, ψ_x^2 and the function $G_{xx}(f)$. Note that $\psi_x^2 = \sigma_x^2 + \mu_x^2$, so that for this particular sequence

$$\psi_x^2 = \sigma_x^2 = \int_0^\infty G_{xx}(f)\, df = \frac{1}{12} \quad - \quad \text{total power (P)} \tag{7.3}$$

If these data are considered to be samples of a continuous signal $x(t)$ with sampling interval Δt , then the power spectrum should look like that shown in figure 7.1.

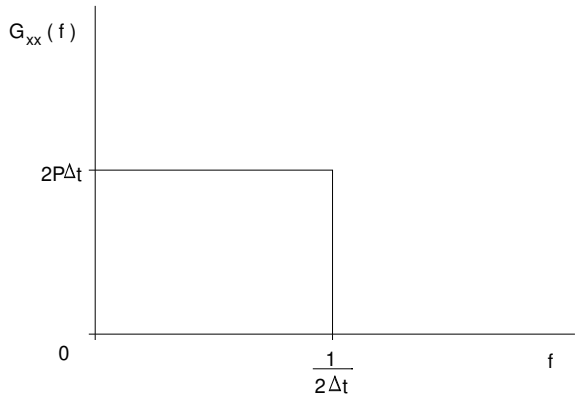

Figure 7.1 White noise spectrum with total power P

For convenience, it is useful to adjust the total power in the sequence in order that the power density $G_{xx}(f) = 1$ for $0 \le f \le f_c$ where $f_c = 1/2\Delta t$ is the Nyquist frequency. To accomplish this, consider the sequence

$$y_n = K\left(x_n - \frac{1}{2}\right) \tag{7.4}$$

where x_n is a random number on the interval from 0 to 1 as before and y_n ranges from $-K/2$ to $K/2$. For this sequence, the mean and variance are

$$\mu_x = 0 \qquad \text{and} \qquad \sigma_x^2 = \frac{K^2}{12} \tag{7.5}$$

Now, choose $P = 1/2\Delta t$, so that

$$\psi_x^2 = \sigma_x^2 = \int_0^\infty G_{xx}(f)\, df = \frac{K^2}{12} = P = \frac{1}{2\Delta t} \tag{7.6}$$

From this, we find that $K = \left(\frac{6}{\Delta t}\right)^{1/2}$, and for this value of K , the power density for the sequence defined in equation 7.4 satisfies

$$G_{yy}(f) = \begin{cases} 1 & 0 \le f \le f_c \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

A plot of this power spectrum is shown in figure 7.2.

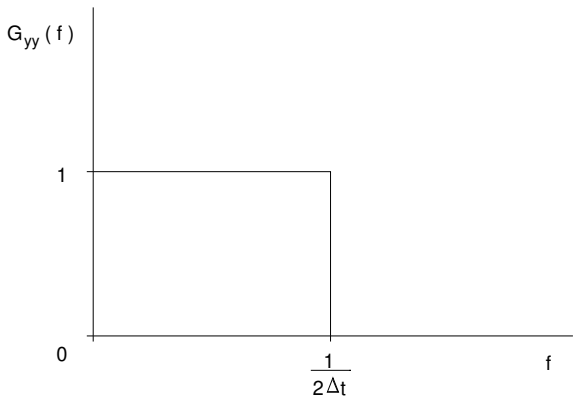

Figure 7.2 White noise spectrum with total power $1/2\Delta t$

The final step is to create a new sequence z_n by passing the values y_n through a bandpass filter with transfer function $H(f)$. The power density for this new sequence is given by

$$G_{zz}(f) = |H(f)|^2 \; G_{yy}(f) \tag{7.8}$$

A recursive digital bandpass filter with $|H(f)| = 1$ and described in equation 3.15 has been used in the COLOR routine. The user specifies the two (-3 dB) points, f_1 and f_2, and the number of cascaded filter stages to use. The order of the filter is twice the number of cascaded stages. The output of the program is a sequence z_n with a power spectrum as shown in figure 7.3.

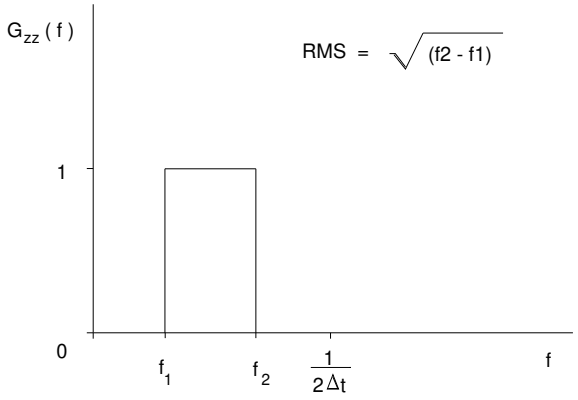

Figure 7.3 Colored noise spectrum with total power $(f_2 - f_1)$

The color routine was used to create the sequence z_n (simulated sampling of the continuous signal $z(t)$) which was stored in a data file that could be read by the spectrum routine. The file contained 100 records with 2048 data points per record. The spacing between data points was 0.01 seconds. The parameters for the bandpass filter were: $f_1 = 20\ Hz$, $f_2 = 30\ Hz$ and five cascaded stages for a tenth order filter. A plot of a portion of the simulated time series appears in figure 7.4. The data file was processed by the spectrum routine several times using various choices of tapering functions, and the power spectra are plotted in figure 7.5. When computing these spectra, no mean removal or detrending was used, and no further filtering was performed. As can be seen, $G_{zz}(f) = 1$ in the passband and rolls off to zero very rapidly elsewhere. (Note that the base 10 logarithm of these numbers are plotted in order to magnify the rolloff region.) This plot clearly shows the advantages gained with the use of tapering and overlapped processing as opposed to no tapering; however, in this case, the choice of tapering function makes little difference. Table 7.1 shows the rms values reported for each case; all of these are within 0.6 % of the theoretical value, and the minor variations are due to the fact that the rolloff is not *perfect*.

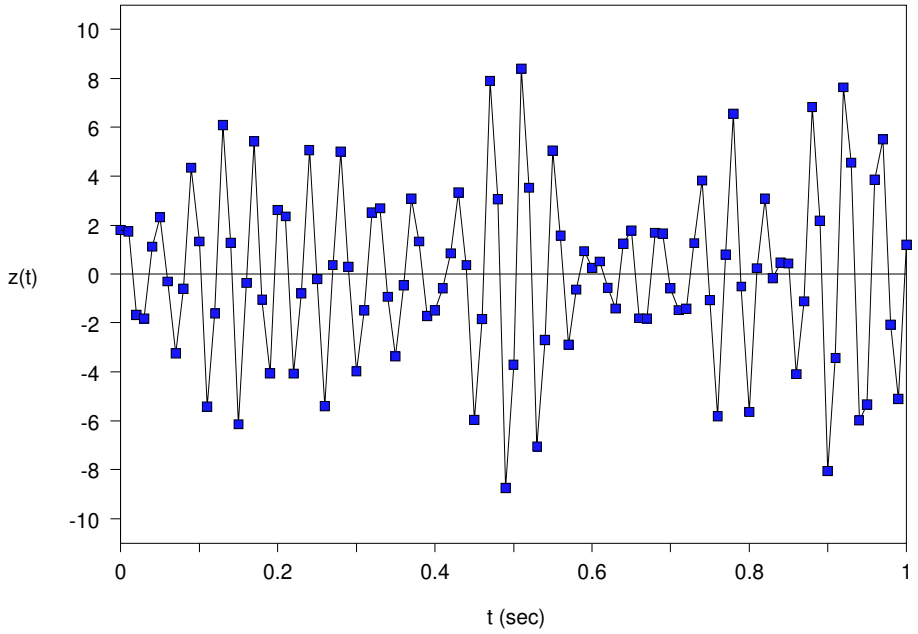

Figure 7.4 Simulated time series $z(t)$

	Theory	No Taper	Hanning	Parzen	Tukey	Welch
rms values	3.16228	3.14463	3.14665	3.14617	3.14315	3.14429
error (%)	--	0.56	0.49	0.51	0.60	0.57

Table 7.1 RMS values reported by spectrum routine

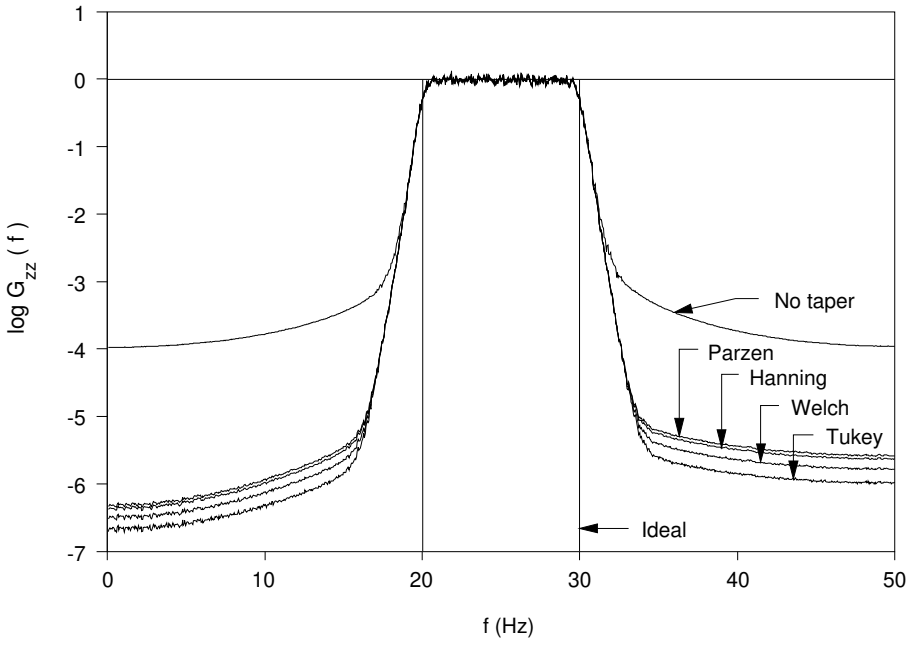

Figure 7.5 Power spectra for colored noise; total power = $\sqrt{10}$

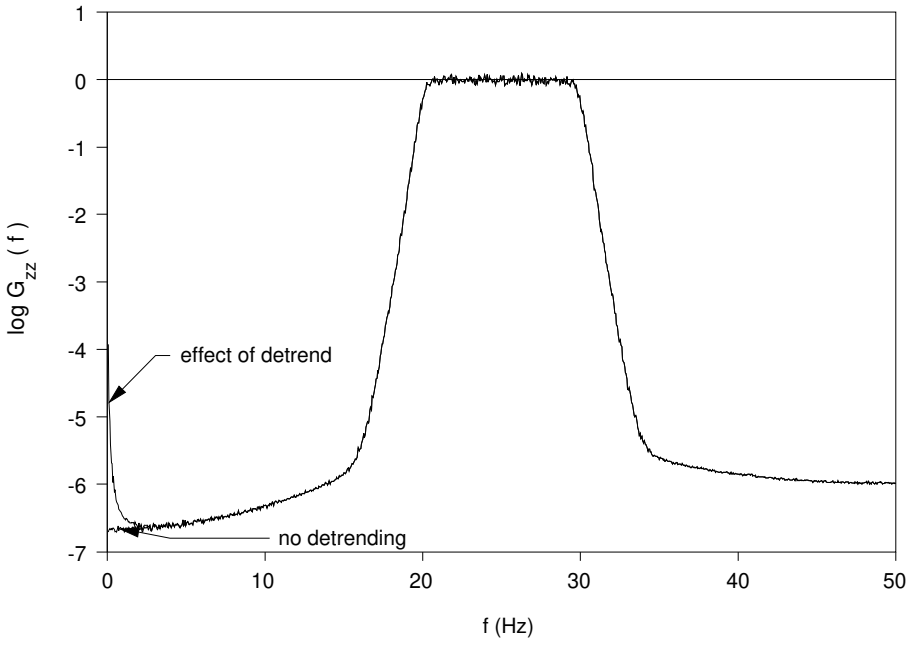

Figure 7.6 Effect of detrend on power spectra

For data generated by the computer, no linear trend in the data is to be expected, and the detrending operation should not be used. The indiscriminate use of this operation can lead to added power at $f = 1/N\Delta t$ since this operation is applied record by record; this can only occur if the power at this frequency is lower than the small amount of noise generated by the use of this operation. In almost every case where the random data originates from an experiment (vis-a-vis computer generated *perfect* random data) this problem will not occur, but the user should be aware of the possibility. A plot of the added power that occurs with the use of the detrending operation is shown in figure 7.6.

The following is an example session showing the various prompts and responses that were used with the color routine for this example.

```
color

Enter the low frequency cutoff : 20.0
                                                    (Defines passband)
Enter the high frequency cutoff : 30.0

Enter delta T seconds. (1.0/Sample Rate) : 0.01     (Sampling rate = 100 Hz)

Enter # of filter sections to cascade : 5           (Tenth order filter)

Enter # of points per record (N) : 2048             (Must be power of two)

Enter # of records (NUMREC) : 100                   (No limit on # records)

Enter a negative integer : -5                       (Required seed value for)
                                                    (random number generator)
Enter output file name (4 chars) : COLR

                    C O L O R E D   N O I S E   U T I L I T Y

                              Creating COLR.DAT now.

                              Writing record 0100

                    Use voltage transformation factor = 20.0 / 4096.0

                         Program terminated successfully.
```

Next is an example session showing the prompts and responses needed to process the COLR.DAT file using the spectrum routine.

```
spectrum colr

Transform into (V)oltages or into (O)ther quantity
or read (R)eal data which is already transformed : v

Enter VOFST 1) 10/4096, 2) 20/4096, 3) 2/4096 or 4) other : 2

Remove (M)ean value, (D)etrend or (N)either : n

Filter the data using (L)owpass, (H)ighpass
                      (B)andpass, Band(S)top or (N)one : n

How many channels of data (1 or 2) : 1

(A)mplitude, (P)ower or (C)ross spectrum : p

Taper the data (Y/N) : y

(H)anning, (P)arzen, (T)ukey or (W)elch window : h

Use overlap (Y/N) : y

                    Processing COLR.DAT now.

          Record 0100   Ave = .15974E-03    Rms = 3.1035

          File Totals :    Ave = .26703E-05    Rms = 3.1446

                    Taper applied = Hanning.
                    50% overlap used.
                    0200 records will be written.

                    Tapering record 0200.

                    Transforming records 0199 and 0200.

                    Record 0200 was discarded.

                    Writing COLRP.PRN now.

                          mean            rms
                     2.18332E-06        3.1467

                    norm. random error = .10000

                    starting time    :   09:15:01.71
                    ending time      :   09:16:06.68
                    elapsed time     :   00:01:04.97

                    Program terminated successfully.
```

Note that the maximum number of points per record is controlled by the value of NMAX. The value of NMAX is currently set to 16384. (If the user should ever desire to make a change to the value of NMAX, the change must be made both in the COLOR routine and also in the SPECTRUM routine. NMAX must be the same number in both routines at all times.)

7.2 Sinusoidal Input

When developing the spectrum routine, it was felt that a simple test would be to attempt to determine the Fourier coefficient (amplitude) of a sinewave by computing its amplitude spectrum. Although this can be done, it is not a trivial procedure and is therefore repeated here as an example. A sinewave generation routine was written and is included on the diskette as SINE; this routine creates a data file that can be read by the spectrum routine. Each record of this data file contains a portion of the time series

$$x(t,k) = A \sin [2\pi f_0 t + \theta_k] \tag{7.9}$$

where A is the amplitude, f_0 is the frequency and θ_k is a phase angle which changes from record to record and is a function therefore of the record number k. The presence of a changing phase angle from record to record is merely to simulate sampling of noncontiguous chunks of the continuous function. This phase angle will not alter the value of the amplitude spectrum and is therefore left out of the following derivation for simplicity.

The finite-range Fourier transform defined in equation 4.2 can be computed for this function $x(t)$ and is given by

$$X(f,T) = \frac{AT}{2i} \left[\frac{\sin \pi (f-f_0) T}{\pi (f-f_0) T} e^{-i\pi (f-f_0) T} - \frac{\sin \pi (f+f_0) T}{\pi (f+f_0) T} e^{-i\pi (f+f_0) T} \right] \tag{7.10}$$

This function must now be evaluated at $f = f_0$, so taking the limit gives

$$\lim_{f \to f_0} X(f,T) = \frac{AT}{2i} \left[1 - \frac{\sin 2\pi f_0 T}{2\pi f_0 T} e^{-i2\pi f_0 T} \right] \tag{7.11}$$

where the first term was evaluated using l'Hôpital's rule. The modulus of this quantity is a function of the record length T, and is given by

$$\left| \lim_{f \to f_0} X(f,T) \right| = \frac{AT}{2} \left[\frac{\sin^2 2\pi f_0 T}{(2\pi f_0 T)^2} - \frac{\sin 4\pi f_0 T}{2\pi f_0 T} + 1 \right]^{\frac{1}{2}} \tag{7.12}$$

To further examine this dependence upon the record length, one can express T in terms of n , the number of cycles of the sinewave, using

$$T = N\Delta t = \frac{n}{f_0} \tag{7.13}$$

With this value for the record length, equation 7.12 becomes

$$\left| \lim_{f \to f_0} X(f,T) \right| = \frac{AT}{2} \left[\frac{\sin^2 2n\pi}{(2n\pi)^2} - \frac{\sin 4n\pi}{2n\pi} + 1 \right]^{\frac{1}{2}} = \frac{AT}{2} f(n) \tag{7.14}$$

The function $f(n)$ is plotted in figure 7.7 and is shown to oscillate about the value one. The extent of the oscillation decreases with increasing n , and in the limit as the record length goes to infinity, the function equals one.

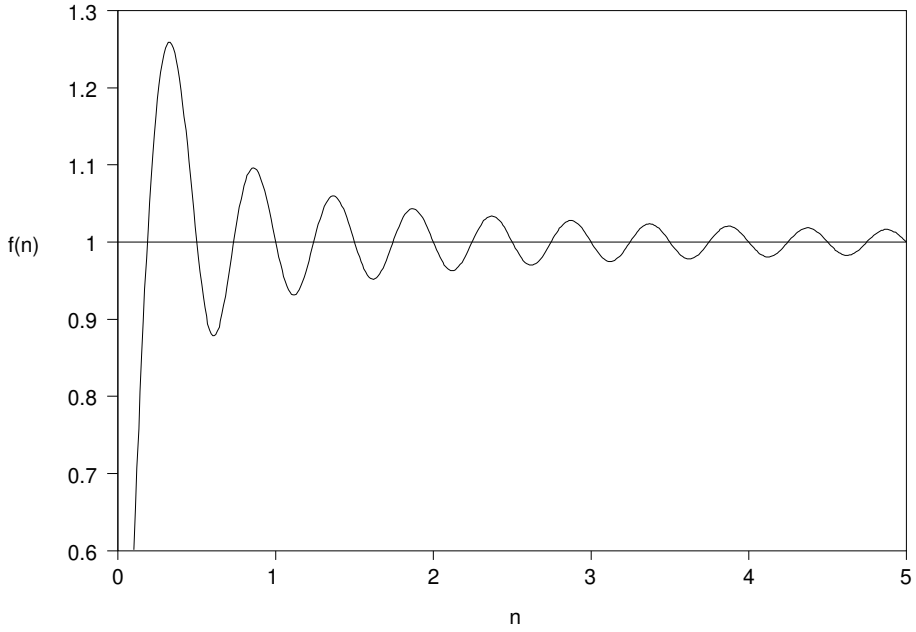

Figure 7.7 Oscillatory function $f(n)$

Due to the finite nature of the signal, one can obtain the limiting value for the Fourier coefficient only by paying careful attention to the choice of record length. Specifically, if the record length is chosen to contain an integral number of cycles of the sinewave, then as can be seen from figure 7.7, $f(n) = 1$ for integer n . Summarizing then, the limiting value

of the amplitude spectrum of the sinewave can be computed if the record length is chosen to contain an integral number of cycles of the sinewave; when this criterion is satisfied, then the modulus is given by

$$| \lim_{f \to f_0} X(f,T) | = \frac{AT}{2} \qquad \text{for integer n} \tag{7.15}$$

Now, using the definition given in equation 4.28, the value of the amplitude spectrum reported by the spectrum routine at the frequency f_0 will be the ensemble average over all records of the data file of the modulus defined in equation 7.15. Three further constraints are placed upon the record length in addition to the requirement of integral values of n : N must be a power of two, N should be large for better accuracy and $\Delta t \leq 1/2f_0$ which is a statement of the fact that the frequency of the sinewave must be less than or equal to the Nyquist frequency determined by the sampling interval. These requirements are satisfied if $\Delta t = 1/2^j f_0$, $N = 2^k$ and $k \gg j$ for integer values of j and k ; for these choices $n = 2^{k-j}$. Generally, the best accuracy is obtained when $n = N/4$ ($j = 2$).

As an example of this technique, the sinewave

$$x(t,k) = 1.25 \ \sin \ [2\pi (128.0)\, t \ + \ \theta_k] \tag{7.16}$$

was created using the SINE routine. The file contained ten records with 8192 points per record. The sampling interval was chosen to be (using $n = N/4$)

$$\Delta t = \frac{1}{(4)(128)} = 0.001953125 \tag{7.17}$$

so that the record contained exactly 2048 cycles of the sinewave. The spectrum should have a peak at a frequency of 128 Hz where the amplitude is given by

$$F_x(128) = \lim_{T \to \infty} E\left[| \lim_{f \to 128} X(f,T) | \right] = \frac{(1.25)(8192)}{(512)(2)} = 10 \tag{7.18}$$

The actual value reported by the spectrum routine was 9.9656 at a frequency of 128.0082 Hz which was the closest discrete frequency value. A plot of the amplitude spectrum is shown in figure 7.8. When computing this spectrum, no mean removal or detrending was used, and no filtering was performed. Furthermore, application of a tapering function would significantly change the expected amplitude spectrum and therefore no tapering was used.

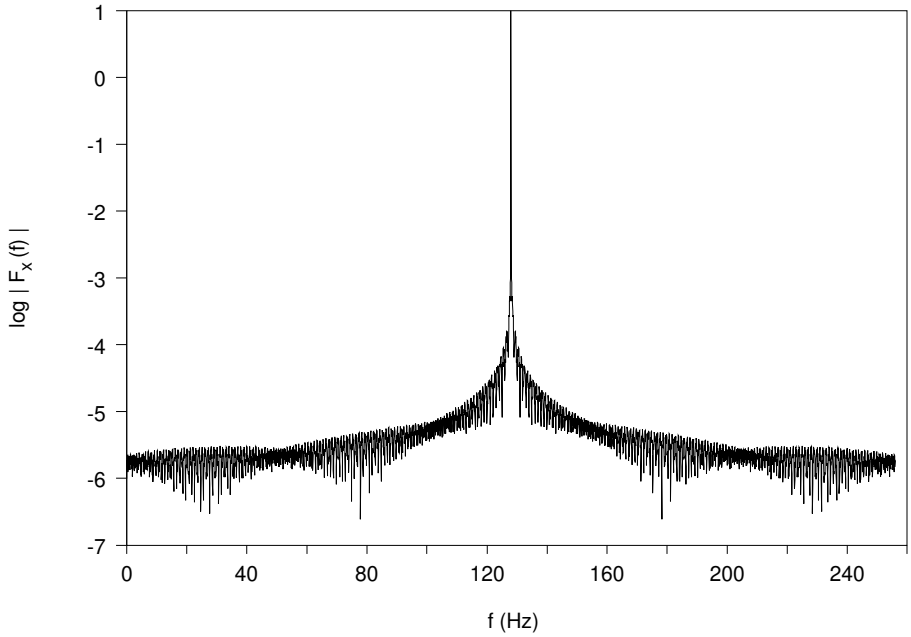

Figure 7.8 Amplitude spectrum of sinewave

The required inputs to the SINE routine follow.

sine

Enter the amplitude of the sinewave : 1.25	(gives a simple answer)
Enter the frequency of the sinewave : 128.0	(can be any desired number)
Enter the phase (in degrees) : 32.8	(choose any initial value)
Enter DC value : 0.0	(no DC for this example)
Enter delta T secs. (1.0/Sample Rate) : 0.001953125	(according to eq. 7.17)
Enter # of points per record (N) : 8192	(must be power of 2)
Enter # of records (ASIZE) : 10	(OK for this example)

Enter output file name (4 chars) : SINE

S I N E W A V E C R E A T I O N U T I L I T Y

Creating SINE.DAT now.

Writing record 0010

Use voltage transformation factor = 20.0 / 4096.0

Program terminated successfully.

The required inputs to the spectrum routine to create the amplitude spectrum given in figure 7.8 follow.

```
spectrum sine

Transform into (V)oltages or into (O)ther quantity
or read (R)eal data which is already transformed : v

Enter VOFST 1) 10/4096, 2) 20/4096, 3) 2/4096 or 4) other : 2

Remove (M)ean value, (D)etrend or (N)either : n

Filter the data using (L)owpass, (H)ighpass
                      (B)andpass, Band(S)top or (N)one : n

How many channels of data (1 or 2) : 1

(A)mplitude, (P)ower or (C)ross spectrum : a

Taper the data (Y/N) : n

                    Processing SINE.DAT now.

               Record 0010     : Ave = .00000  Rms = .88208

               File Totals     : Ave = .00000  Rms = .88090

                    No tapering applied.
                    0010 records will be used.

                    Transforming records 0009 and 0010.

                    Writing SINEA.PRN now.

                    starting time    :   20:24:53.77
                    ending time      :   20:25:13.21
                    elapsed time     :   00:00:19.44

                    Program terminated successfully.
```

Note that the maximum number of points per record is controlled by the value of NMAX. The value of NMAX is currently set to 16384. (If the user should ever desire to make a change to the value of NMAX, the change must be made both in the SINE routine and also in the SPECTRUM routine. NMAX must be the same number in both routines at all times.)

7.3 Experimental Noise Input

The input data file for this example was derived from an experiment employing an electrodynamic shaker table. This device is essentially a large solenoid mated to an electronic feedback system which is used to accurately control the oscillatory motion of the core. The displacement amplitude and frequency can be set to desired levels and then maintained very accurately. This instrument was used in an experiment designed to test the frequency response of an optical system which was built to measure displacement of a surface at a point. Prior to the actual frequency response test, the optical system was used to measure the displacement of the core of the solenoid with the shaker table energized but undergoing no motion. The computation of a power spectrum of this displacement *noise* can then be a useful diagnostic tool for the identification of noise sources in subsequent displacement spectrum measurements.

The signal was digitized using a 12-bit A/D converter with an input range of -10 to 10 V. The sampling rate was 2 kHz which resulted in a sampling interval of $\Delta t = 0.0005$ seconds. 80 records of data were acquired with 2048 points per record. The power spectrum was computed for a range of frequencies up to the Nyquist frequency which was 1 kHz; the resolution was given by $\Delta f = 1/N\Delta t = 0.97656\ Hz$. All power spectra shown in this section were computed by first detrending, then tapering the input time records with the Welch taper and employing 50% overlap. The resulting power spectrum is shown in figure 7.9; logarithmic scales are used for greater clarity. The figure reveals that most of the power in the signal lies below 100 Hz. A 60 Hz contribution attributable to the power supply may be identified as well as other resonances and their harmonics.

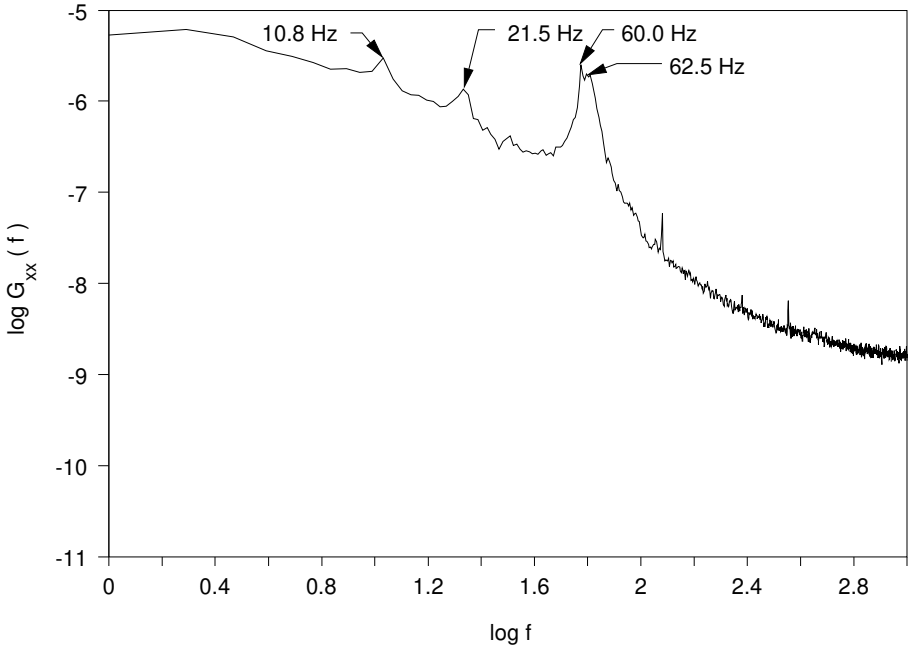

Figure 7.9 Power spectrum of displacement *noise*

This input file allows a useful demonstration of the filtering capabilities of the spectrum routine. Shown in Figure 7.10 are power spectra of the same input file for which figure 7.9 was obtained; however, prior to the spectral calculation, each of the four types of filters available for use were applied to the data. In each case five cascaded filter stages were used providing a tenth order digital Butterworth filter. As can be seen, the rolloff is extremely rapid with power in the stopband 3 to 4 decades below that in the passband. This is an effective means for removing undesirable frequency components from the data prior to spectral computation. This can also be useful for quickly estimating the amount of power contributed by specific peaks or regions of the spectrum. The rms value, calculated as the square root of the area under the spectral density curve (according to eq. 4.13), is given in table 7.2 for each of the filter cases shown in figure 7.10 and for the no-filter case plotted in figure 7.9.

	No filter	Lowpass	Highpass	Bandpass	Bandstop
rms values	0.014803	0.013319	0.009529	0.005894	0.010116

Table 7.2 RMS values reported after application of each filter type

The table indicates that about 10% of the contribution to the root-mean-square value of the signal may be found above 100 Hz, about a third lies below 30 Hz and just under half may be found between 50 and 100 Hz. However, adding the rms value for the bandpass case to the rms value for the bandstop case does not equal the rms value for the no filter case. This discrepancy is due to the fact that *some* power is in the stopband for the bandpass and bandstop cases, and this contributes a small amount to the rms values indicated. Therefore, this method should only be used for rough estimates; the most accurate method is to integrate the spectral density function over frequency bands of interest.

Substitute Figure 7.10 for this page.

7.4 Sampling Considerations

A full discussion of sampling procedures will not be reiterated here; a complete discussion of these techniques can be found in the references, and the interested reader is urged to look there. Instead, this section attempts to present some of the considerations necessary when making choices among the various sampling parameters, and the manner in which these choices affect the spectral density results.

The most important parameter is probably the choice of sampling interval Δt. In general, one attempts to choose this quantity to be of the order of the smallest time scale of interest in the physical phenomenon under investigation. The selection of Δt determines the largest frequency (bandwidth) of the resulting spectral analysis. This critical frequency f_c is known as the Nyquist frequency and is given by

$$f_c = \frac{1}{2\Delta t} \tag{7.19}$$

Another quick way for determining the Nyquist frequency is to note that $f_c = f_s/2$ where $f_s = 1/\Delta t$ is the sampling rate at which data are acquired. If energy exists in the signal above the Nyquist frequency determined by the desired sampling rate, then this energy must be removed by means of an analog lowpass filter prior to A/D conversion. This is necessary to prevent *aliasing* of the high frequency components in the data. Aliasing is essentially confusion among the high and low frequency components in the data that occurs whenever the sampling rate is set too low. A very good discussion of aliasing may be found in Bendat and Piersol (1986).

Another parameter of interest is the number of data points per record per channel N. This quantity combined with the choice of Δt determines the record length $T = N\Delta t$ and the resolution bandwidth of the spectral analysis,

$$\Delta f = \frac{1}{N\Delta t} \tag{7.20}$$

A large value of N for a given value of Δt gives a smaller Δf and can give a more detailed spectral density estimate. On the other hand, if the record length T is very long, *realizability* problems may occur: is the physical phenomenon *stationary* for these long time periods? Is there a limitation on maximum file size? Can the resulting massive amounts of data be adequately stored?

Another important parameter is the number of records n_r of length T to sample for each data file. These records should be contiguous in time (in order to use overlapping) and constitute an ensemble of statistically stationary records; the expected value operation

contained in the definitions of the spectral density estimates is approximated when an ensemble average is calculated for this set. Recall, that it is n_r (the number of records *before* overlapping) which is used in the determination of the errors associated with each of the spectral density estimates and not N or Δt. Increasing n_r rapidly decreases the normalized random error ε up to about $n_r = 100$ or so; for larger values of n_r the error decreases much more slowly. There may be certain situations in which, for a given overall file size, it is better to increase n_r and decrease N for a given Δt in order to increase accuracy at the expense of resolution bandwidth Δf.

8 Utility Routines

Four utility programs are included on the diskette. The SINE and COLOR routines have been discussed in the previous section concerning examples. MAKDAT is a routine designed to read a data file in ASCII format and then to write the necessary file header and data to an unformatted file. MAKCOEF is similar; it reads a user-prepared ASCII format coefficient data file and then writes the file header and coefficients to an unformatted file. It is easiest, of course, to add the necessary code to create the unformatted files directly to the routine that is used to create (or collect) the data in the first place; however, if the user is unfamiliar with unformatted FORTRAN data files or the FORTRAN language in general, then these routines attempt to solve the problem.

8.1 MAKDAT routine

In order to use MAKDAT, the data to be analyzed by the spectrum routine must be generated, or collected by an A/D converter, and then stored to a file in ASCII format. In this format the file may be viewed by any ASCII editor, can be displayed on the screen and can be printed on a printer. This is not an optimum way to store data; these files tend to be extremely large. In anticipation of this fact other types of data files were explored for possible use, and unformatted FORTRAN files appeared to be the simplest solution. Unformatted files are typically a factor of five smaller in size than the equivalent ASCII file. An example will be given to illustrate the means for transferring the data to unformatted files which can then be used by SPECTRUM.

Suppose that 100 records of one channel of data saved as 2-byte integers and sampled at 10000 samples per second with 2048 data points per record have been acquired. These data should be placed in an ASCII file with the file extension [.ASC] and with a file name conforming to the various naming conventions discussed earlier. The file must contain numbers only; the numbers may be written one per line or many per line (but this does not shorten the length of the file). Blank lines may be added if desired. The first number in the file should be the first data point of the first record; the 2049th number in the file should be the first data point of the second record, and the last number in the file should be the 2048th data point of the 100th record. A session with MAKDAT to accomplish this transformation would proceed as follows.

```
makdat

(I)nteger (2-byte) or (F)loating-point (4-byte) data : i

Enter # of channels (1 or 2) : 1

One channel     : Total # points per record <= 16384.
Two channels    : Total # points per record per channel <= 8192.

Enter # of points per record (power of two) : 2048

One channel     : Must be EVEN # of records.
```

```
Two channels   : May be EVEN or ODD # of records.

Enter # of records in the data file : 100

One chan     :  Delta t is spacing between data points.
Two chans    :  Delta t is spacing between data pts - SAME channel
                Delta t divided by 2 is spacing between data pts
                - different channels.

Enter sampling interval delta t (secs) : 0.0001

Enter ASCII input file name (4 chars) : test

                D A T A   F I L E   C R E A T I O N   U T I L I T Y

                        Creating TEST.DAT now.

                        # channels = 1
                        record size = 4096 bytes
                        # of records = 100
                        delta t = 100 microseconds

                        Record 0100

                        Program terminated successfully.
```

The MAKDAT routine reads the user-prepared TEST.ASC file and calculates the values needed for the file header. The file header would consist of

```
1       4096       100       100
```

The routine then transfers the data to the unformatted file with the name TEST.DAT. Finally, to process this file with the spectrum routine the user would enter: spectrum test.

For two channels of data, it is necessary that the numbers be stored in each record of the ASCII file in the following manner:

one channel: $x_0\, x_1\, x_2\ \ldots\ x_{N-1}$

two channels: $x_0\, y_0\, x_1\, y_1\, x_2\, y_2\ \ldots\ x_{N-1}\, y_{N-1}$

using the notation defined earlier. Therefore, the first number in the file will be the first data point of the first record for channel 1; the second number will be the first data point of the first record for channel 2; the third number in the file will be the second number for the first record of channel 1, and so on. Also, care should be taken to enter correct responses to the prompts when two channels of data are to be transferred. In response to the number of points per record per channel prompt, the user would enter 1024 points if the data file

consisted of two channels of data with 2048 points per record and therefore 1024 points for each channel in each record. Note that the maximum number of points per record is controlled by the value of NMAX; the maximum number of points per record per channel is then NMAX / 2. The value of NMAX is currently set to 16384. (If the user should ever desire to make a change to the value of NMAX, the change must be made both in the MAKDAT routine and also in the SPECTRUM routine. NMAX must be the same number in both routines at all times.) Similarly, when responding to the prompt for the sampling interval, the user would enter 0.0002 seconds for the sampling interval between data points of the *same* channel for the case in which the data were sampled at 10000 samples per second. Although, 0.0001 seconds is the sampling interval between adjacent data points in the record, the sampling interval for data points of the same channel is a number twice as large.

8.2 MAKCOEF routine

The primary reason for using unformatted coefficient data files with the spectrum routine is to allow the user a means for preparing all of the coefficient data for the transformation of voltages into other dimensional quantities for several files which will then be processed by SPECTRUM in a batch. The MAKCOEF utility allows the user to prepare this coefficient data in an ASCII file; the utility will write the file header and then transfer the data to an unformatted coefficient data file which can be read by the spectrum routine.

Suppose that it was desired to process the data files D061.DAT thru D086.DAT in a batch by the spectrum routine. Each of these data files consisted of two channels of 2-byte integers taken from an experiment in fluid mechanics, say. Furthermore, suppose that the quantities sampled were two components of a fluid velocity, which were transduced into voltages by some means and then sampled by an A/D converter. The user knows the calibration data for the transducer, and the required transformation from voltages back into velocities can be accomplished for each velocity component by using a polynomial of up to fourth degree in the form:

$$u_n = b_0 + b_1 v_n + b_2 v_n^2 + b_3 v_n^3 + b_4 v_n^4 \tag{8.1}$$

where v_n is the nth data point expressed as a voltage, $b_0, b_1, \ldots, b_4$ are the coefficients of the velocity transformation and u_n is the nth data point which has been converted to a velocity. Now, suppose further that the data in each of the files were taken over a period of time in which the calibration coefficients changed; thus, a different set of coefficients are required for each of the 26 files to be processed. Following the instructions in the section on coefficient data files, this scenario would require the creation of two unformatted coefficient data files, DCON.DAT and DCON2.DAT, each containing the 26 sets of five coefficients $b_0, b_1, \ldots, b_4$ for the velocity transformation for channel 1 and channel 2, respectively.

The user should create each ASCII data file with a four character name followed by the extension [.ASC]. The names should conform to the various naming conventions specified earlier. The ASCII file must contain numbers only, although blank lines may be inserted into the file if desired. The numbers will be transferred to 4-byte floating point quantities in the unformatted data file thereby limiting the coefficients to single precision (6 or 7 decimal places). The coefficients may be stored one per line or many per line in the ASCII file. The first number in the file will be b_0 for channel 1 of data file D061.DAT; the fifth number in the file should be b_4 for channel 1 of data file D061.DAT; the sixth number in the file will be b_0 for channel 1 of data file D062.DAT, and so on. The 130th and last number in the file should be b_4 for channel 1 of data file D086.DAT. The user should then create a second ASCII file in a similar manner specifying all of the coefficients for channel 2 of the 26 files to be processed. The utility MAKCOEF should then be executed twice, first transferring the contents of the first ASCII file (for channel 1) to DCON.DAT and then for the second ASCII file (for channel 2) to DCON2.DAT. A session using MAKCOEF would proceed as follows.

```
makcoef

Enter first letter of data file to
which these coefficients will be associated : d

Are these coefficients for channel (1 or 2) : 1

Enter # of sets of coefficients : 26

Enter starting file number : 61

Enter ASCII input file name (4 chars) : chn1

                C O E F F I C I E N T   F I L E   C R E A T I O N   U T I L I T Y

                        Creating DCON.DAT now.

                        # sets of coeffs = 26
                        # coeffs in each set = 5
                        # of starting file = 61

                        Program terminated successfully.
```

The MAKCOEF routine will read the ASCII file CHN1.ASC and will write the following file header to the unformatted file DCON.DAT:

```
26      61
```

The coefficients for channel 1 are then transferred to the unformatted file. Note that 5 coefficients per set are required; therefore, this number is not written to the file header. If, at a later time, the user wishes to reprocess the files D074.DAT through D080.DAT (perhaps specifying different options in the spectrum routine), it is not necessary to re-create the two coefficient data files; as long as the data files to be reprocessed are chosen from among the 26 original members of the set, the spectrum routine will use only the coefficients that it needs. Similarly, if the user wishes to process the data files in an arbitrarily numbered order (not consecutively), the coefficient data files do not need to be changed as long as the data files come from the 26 original members of the set. Thus, D061.DAT, D064.DAT, D069.DAT and D086.DAT could be processed by SPECTRUM without changing the coefficient data files. The converse is also true, however. If the user desires at the outset to process only the four arbitrarily ordered files above, two coefficient data files must be created which contain not 4, but 26 sets of 5 coefficients for the data files D061.DAT through D086.DAT. The spectrum routine requires that sets of coefficients in the coefficient data file be listed for consecutively numbered files even if not all of the sets of coefficients are used. With this in mind, it is perhaps best to create the coefficient data files for an entire set of input data files; then members of the set can be processed by the spectrum routine either consecutively or in an arbitrarily numbered order at any subsequent time. Of course, the user could transform the voltages back into dimensional quantities using his/her own code and store the resulting floating-point numbers into ASCII data files (then use MAK-DAT) or directly into unformatted data files to be processed by SPECTRUM; then, all of this mess concerning coefficient data files could be avoided altogether.

Note that the maximum number of points per record is controlled by the value of NMAX. The value of NMAX is currently set to 16384. The maximum number of files that may be processed by SPECTRUM in a batch is controlled by the setting of IFMAX; this is currently set to 100. (If the user should ever desire to make a change to the value of NMAX or IFMAX, the change must be made both in the MAKCOEF routine and also in the SPECTRUM routine. NMAX and IFMAX must be the same numbers in both routines at all times.)

9 References

Bendat, J.S. and Piersol, A.G. 1986 *Random Data*, 2nd ed., John Wiley & Sons, Inc., New York.

Hess, D.E. 1990 An experimental investigation of a compliant surface beneath a turbulent boundary layer. Ph.D. thesis, Johns Hopkins University.

Microsoft Fortran Optimizing Compiler, Version 5.00, Microsoft Corporation, Redmond, Washington, 1989.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. 1986 *Numerical Recipes*, Cambridge University Press, New York.

Sirivat, A. 1985 *Statistical Package for the Analysis of Data (SPAD)*, private communication.

Stearns, S.D. 1975 *Digital Signal Analysis*, Hayden Book Co., Rochelle Park, New Jersey.